Volume 17 Number 3

ACTA CYBERNETICA

Szeged, 2006

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. Fifty reprints are supplied for each article published.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage http://www.inf.u-szeged.hu/actacybernetica/ .

János Demetrovics
MTA SZTAKI
Budapest, Hungary
demetrovics@sztaki.hu

Bálint Dömölki
IQSOFT
Budapest, Hungary
domolki@iqsoft.hu

Zoltán Ésik
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
ze@inf.u-szeged.hu

Zoltán Fülöp
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
fulop@inf.u-szeged.hu

Ferenc Gécseg
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
gecseg@inf.u-szeged.hu

Jozef Gruska
Institute of Informatics/Mathematics
Slovak Academy of Science
Bratislava, Slovakia
gruska@savba.sk

Balázs Imreh
Department of Applied Informatics
University of Szeged
Szeged, Hungary
imreh@inf.u-szeged.hu

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
helmut@csd.uwo.ca

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
vogler@inf.tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Editorial

Zoltán Fülöp has decided to step down as Managing Editor. He has been running the Journal since 1993. Over the years, the quality of scientific content has been kept as high as possible. Much of this success is due to his enormous dedication to the Journal: He has been coordinating the review process of more than 200 regular papers covering a wide range of topics in Computer Science and co-edited several special issues: *Conference of PhD Students in Computer Science* 1998, 2000, 2002, and 2004; *Symposium on Programming Languages and Software Tools* 2001; and a special issue *Dedicated to Professor Ferenc Gécseg on the occasion of his 60th birthday* in 1999. The main goal he set out for himself, namely to publish high quality original papers in the field of Computer Science, has been reached: our Journal is reviewed by *Mathematical Reviews*, *Computing Reviews*, *Zentralblatt für Mathematics* and it is indexed by *DBLP*. I would like to take this opportunity to thank him for his excellent work. In the future, Zoltán will continue to support us as an Editor.

Running such a journal takes time and dedication. I talked to a number of colleagues to see whether they would be willing to take over the load that Zoltán Fülöp has been carrying for such a long time. I'm pleased to announce that Zoltan Kato has accepted to become Managing Editor. He is a recognized researcher in the field of Computer Vision and has the necessary experience in editorial work (currently he is on the editorial board of *IEEE Transactions on Image Processing*). I know he will continue Zoltán Fülöp's efforts in increasing the scientific impact of the Journal.

Our success, however, will always critically depend on continuous commitment from all of us: authors who contribute high-quality manuscripts, reviewers who respond timely with expert recommendations, editors who take sound decisions, and readers who use and refer others to our published papers. With the new editorial board, I invite all of you to help keep this Journal exciting and efficient, so that everyone in the community can benefit.

János Csirik
Editor-in-Chief

Acta Cybernetica **17** (2006) 449–457.

On the closedness of nilpotent DR tree languages under Boolean operations*

F. Gécseg† and Gy. Gyurica†

To Professor László Leindler on his 70th birthday

Abstract

This note deals with the closedness of nilpotent deterministic root-to-frontier tree languages with respect to the Boolean operations union, intersection and complementation. Necessary and sufficient conditions are given under which the union of two deterministic tree languages is also deterministic. The paper ends with a characterization of the largest subclass of the class of nilpotent deterministic root-to-frontier tree languages closed under the formation of complements.

1 Introduction

In [3] we introduced nilpotent DR tree languages and characterized them by means of syntactic monoids. For string languages there is another well known characterization of nilpotency: a language L is nilpotent if and only if L or the complement of L is finite. Obviously, this implies that the complements of nilpotent languages are also nilpotent. This result is true for tree languages recognized by nilpotent frontier-to-root tree recognizers (see, [2]), but it does not hold for nilpotent DR tree languages. In this note we study the closedness of nilpotent DR tree languages under the Boolean operations: union, intersection and complementation. We introduce the concepts of union and intersection direct products of DR tree recognizers, which turn out to be very useful in studying unions and intersections of deterministic tree languages. We give necessary and sufficient conditions under which the union of two deterministic tree languages is also deterministic. Moreover, we determine that subclass of the class of nilpotent DR tree languages which is closed under complementation. It will turn out that unary tree languages play an important role in these classes.

Deterministic tree languages have been intensively studied by E. Jurvanen. In [5] she gives several counter examples, among others, for the closedness of deterministic tree languages under union.

*This work has been supported by the Hungarian National Foundation for Scientific Research, Grant T 048786.

†Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary.

2 Notions and notations

In this paper Σ is always a *ranked alphabet*, i.e. a finite nonempty set of operation symbols, and for each $m \geq 1$, we denote by Σ_m the set of m-ary symbols in Σ. We assume that there are no nullary symbols in Σ, but instead a finite non-empty *leaf alphabet* X is used. The set $T_\Sigma(X)$ of Σ-*terms* over X is the least set such that

(1) $X \subseteq T_\Sigma(X)$, and

(2) $\sigma(p_1, \ldots, p_m) \in T_\Sigma(X)$, whenever $m \geq 1$, $\sigma \in \Sigma_m$ and $p_1, \ldots, p_m \in T_\Sigma(X)$.

Such terms are regarded as trees in the usual way and we call them ΣX-*trees* (or just *trees*). A ΣX-*tree language* is any subset of $T_\Sigma(X)$.

A (finite) *DR* Σ-*algebra* consists of a non-empty (finite) set A and a Σ-indexed family of *root-to-frontier operations*

$$\sigma^{\mathcal{A}} : A \longrightarrow A^m \quad (\sigma \in \Sigma),$$

where the arity m is that of $\sigma (\in \Sigma_m)$. We write simply $\mathcal{A} = (A, \Sigma)$. A *DR* ΣX-*recognizer* is a system $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$, where $\mathcal{A} = (A, \Sigma)$ is a finite DR Σ-algebra, $a_0 \in A$ is the *initial state*, and $\alpha : X \to \wp A$ is the *final state assignment*. ($\wp A$ denotes the power set of A.)

We extend α to a mapping $\alpha_{\mathcal{A}} : T_\Sigma(X) \to \wp A$ in the following way:

(1) $x\alpha_{\mathcal{A}} = x\alpha$ for each $x \in X$,

(2) $p\alpha_{\mathcal{A}} = \{a \in A \mid \sigma^{\mathcal{A}}(a) \in p_1\alpha_{\mathcal{A}} \times \ldots \times p_m\alpha_{\mathcal{A}}\}$ for $p = \sigma(p_1, \ldots, p_m)$.

The tree language *recognized* by $\mathbf{A}$ is defined as the set

$$T(\mathbf{A}) = \{p \in T_\Sigma(X) \mid a_0 \in p\alpha_{\mathcal{A}}\}.$$

A ΣX-tree language is *DR-recognizable* if it is recognized by some DR ΣX-recognizer.

The *path alphabet* $\hat{\Sigma}$ associated with a ranked alphabet Σ is defined by

$$\hat{\Sigma} = \bigcup_{m>0} \Sigma_m \times \{1, \ldots, m\}.$$

Any element (σ, i) of $\hat{\Sigma}$ is regarded as a letter of an ordinary alphabet, and for convenience we write it as σ_i. Words over $\hat{\Sigma}$ are used for representing paths leading from the root to a leaf in a ΣX-tree. In a letter σ_i appearing in such a representation, the component σ gives the label of a node while the i indicates the direction taken at that node.

For any $x \in X$, the set $g_x(p)$ of x-*paths* in a given ΣX-tree p is defined as follows:

(1) $g_x(x) = \{e\}$, where e is the empty word;

(2) $g_x(y) = \emptyset$ for $y \in X$, $y \neq x$;

(3) $g_x(p) = \sigma_1 g_x(p_1) \cup \ldots \cup \sigma_m g_x(p_m)$ for $p = \sigma(p_1, \ldots, p_m)$.

The mappings g_x are extended to ΣX-tree languages in the natural way, and for any $T \subseteq T_\Sigma(X)$ and $x \in X$, we write $T_x = g_x(T)$. These sets $T_x \subseteq \hat{\Sigma}^*$ are called the *path languages* of T. A ΣX-tree language T is said to be *closed* if $p \in T$ for any ΣX-tree p such that $g_x(p) \subseteq T_x$ for every $x \in X$. As shown in [1] and in [7], a regular tree language is DR-recognizable iff it is closed.

The following result is from [4].

Theorem 1. *For any closed ΣX-tree language the following conditions are equivalent:*

(1) $T \in \mathrm{DRec}_\Sigma(X)$;

(2) *there is a congruence on $\hat{\Sigma}^*$ of finite index saturating all of the path languages T_x $(x \in X)$;*

(3) μ_T *is of finite index.*

The quotient monoid $PM(T) = \hat{\Sigma}^*/\mu_T$ is called the *syntactic path monoid* of $T(\subseteq T_\Sigma(X))$. As usual, set $\hat{\Sigma}^+ = \hat{\Sigma}^* \backslash \{e\}$, and denote by the same μ_T the restriction of μ_T to $\hat{\Sigma}^+$. Then $PS(T) = \hat{\Sigma}^+/\mu_T$ is called the *syntactic path semigroup* of T. Immediately from Theorem 1 one gets

Corollary 2. *A closed tree language is DR-recognizable iff its syntactic path monoid is finite.*

Let $\mathcal{A} = (A, \Sigma)$ be a DR Σ-algebra, $a \in A$ an element and $p \in T_\Sigma(X)$ a tree. Define the word $\overline{\mathrm{fr}}(ap) \in A^*$ in the following way:

1) if $p = x \in X$ then $\overline{\mathrm{fr}}(ap) = a$,

2) if $p = \sigma(p_1, \ldots, p_l)$ then $\overline{\mathrm{fr}}(ap) = \overline{\mathrm{fr}}(a_1 p_1) \ldots \overline{\mathrm{fr}}(a_l p_l)$, where $(a_1, \ldots, a_l) = \sigma^{\mathcal{A}}(a)$.

For a ΣX-tree p set $\mathrm{mh}(p) = \min\{|u| : u \in \bigcup(g_x(p) : x \in X)\}$, where $|u|$ denotes the length of u. In words, $\mathrm{mh}(p)$ is the length of the shortest path leading from the root of t to a leaf.

A DR Σ-algebra $\mathcal{A} = (A, \Sigma)$ is *nilpotent* if there are an integer $k \geq 0$ and an element $\bar{a} \in A$ such that for all $a \in A$ and $p \in T_\Sigma(X)$ with $\mathrm{mh}(p) \geq k$, $\overline{\mathrm{fr}}(ap) = \bar{a}^l$ for a natural number l. This $\bar{a}$ is the *nilpotent element* of $\mathcal{A}$ and k is called the *degree of nilpotency* of $\mathcal{A}$. A DR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ is *nilpotent* if $\mathcal{A}$ is nilpotent. Moreover, a ΣX-tree language T is *nilpotent* if it can be recognized by a nilpotent DR ΣX-recognizer.

A semigroup S is nilpotent if it has a zero-element 0 and there is a non-negative integer k such that $s_1 \ldots s_k = 0$ for all $s_1, \ldots, s_k \in S$. The integer k is the *degree of nilpotency* of S.

For notions and notations not defined here, see [3] and [4].

3 Union

Let $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Sigma)$ be DR Σ-algebras. Their *direct product* $\mathcal{A} \times \mathcal{B} = (A \times B, \Sigma)$ is given by

$$\sigma^{\mathcal{A}\times\mathcal{B}}((a,b)) = ((\pi_1(\sigma^{\mathcal{A}}(a)), \pi_1(\sigma^{\mathcal{B}}(b))), \ldots, (\pi_m(\sigma^{\mathcal{A}}(a)), \pi_m(\sigma^{\mathcal{B}}(b))))$$

($\sigma \in \Sigma_m$, $(a,b) \in A \times B$). Take two DR ΣX recognizers $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ and $\mathbf{B} = (\mathcal{B}, b_0, \beta)$. Their *union direct product* is

$$\mathbf{A} \times^{\cup} \mathbf{B} = (\mathcal{A} \times \mathcal{B}, (a_0, b_0), \alpha \times^{\cup} \beta),$$

where $(\alpha \times^{\cup} \beta)(x) = (\alpha(x) \times B) \cup (A \times \beta(x))$ $(x \in X)$.

Theorem 3. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two normalized DR* ΣX*-recognizers. Then* $T(\mathbf{A}) \cup T(\mathbf{B})$ *is deterministic if and only if* $T(\mathbf{A}) \cup T(\mathbf{B}) = T(\mathbf{A} \times^{\cup} \mathbf{B})$.

Proof. Assume that $T(\mathbf{A}) \cup T(\mathbf{B})$ is deterministic. Observe that

$$T(\mathbf{A}) \cup T(\mathbf{B}) \subseteq T(\mathbf{A} \times^{\cup} \mathbf{B})$$

holds for arbitrary DR ΣX-recognizer $\mathbf{A}$ and $\mathbf{B}$. Take a $p \in T(\mathbf{A} \times^{\cup} \mathbf{B})$. Then, by the definition of the union product, using the assumption that $\mathbf{A}$ and $\mathbf{B}$ are normalized, we obtain that $g_x(p) \subseteq g_x(T(\mathbf{A}) \cup T(\mathbf{B}))$ for each $x \in X$. Therefore, p is in the closure of $T(\mathbf{A}) \cup T(\mathbf{B})$. However, since $T(\mathbf{A}) \cup T(\mathbf{B})$ is deterministic, it is closed, i.e it coincides with its closure. Therefore, $p \in T(\mathbf{A}) \cup T(\mathbf{B})$.

The converse statement is obvious, since $\mathbf{A} \times^{\cup} \mathbf{B}$ is deterministic. □

We show that in order to study whether the union of two given nilpotent DR ΣX-languages is nilpotent, it is enough to check if their union is deterministic. For this, we need

Lemma 4. *Let* $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ *be a nilpotent DR* ΣX*-recognizer. There exists a normalized nilpotent DR* ΣX*-recognizer* $\mathbf{B} = (\mathcal{B}, b_0, \beta)$ *with* $T(\mathbf{A}) = T(\mathbf{B})$.

Proof. Assume that $\mathbf{A}$ is nilpotent of degree k with the nilpotent element $\bar{a}$. Normalize $\mathbf{A}$ in the following way: if $\sigma(a)$ $(\sigma \in \Sigma,\ a \in A)$ contains a 0-state and $\bar{a}$ is a 0-state then replace it by $\sigma(a) = (\bar{a}, \ldots, \bar{a})$. (Observe that none of the states is a 0-state if $\bar{a}$ is not a 0-state.) Let us denote by $\mathbf{A}^* = (\mathcal{A}^*, a_0, \alpha)$ the resultant recognizer. Then $\mathbf{A}^*$ is normalized, deterministic and $T(\mathbf{A}^*) = T(\mathbf{A})$ (see, p. 115 in [4]). It remains to show that $\mathbf{A}^*$ is nilpotent. It is enough to deal with the case when $\bar{a}$ is a 0-state. Let $a \in A$ be a state and p a tree with $\mathrm{mh}(p) \geq k$. The computing of p in $\mathbf{A}$ and $\mathbf{A}^*$ starting in a coincides up to the point when $\mathbf{A}$ arrives at a 0-state. At this node $\mathbf{A}^*$ will be in state $\bar{a}$ and it remains there during the computing of the subtree belonging to this node. Therefore, $\mathbf{A}^*$ is nilpotent also of degree k with the nilpotent element $\bar{a}$. □

Theorem 5. *Let* $S, T \subseteq T_\Sigma(X)$ *be two nilpotent DR tree languages. Then* $S \cup T$ *is nilpotent if and only if it is deterministic.*

Proof. If $S \cup T$ is nilpotent then it is deterministic by definition.

Conversely, assume that $S \cup T$ is deterministic. Let $S = T(\mathbf{A})$ and $T = T(\mathbf{B})$ where $\mathbf{A}$ and $\mathbf{B}$ are normalized nilpotent DR recognizers such that the degree of nilpotency of $\mathbf{A}$ is k and that of $\mathbf{B}$ is l. By Lemma 4, such $\mathbf{A}$ and $\mathbf{B}$ exist. Moreover, by Theorem 3, $S \cup T = T(\mathbf{A} \times^{\cup} \mathbf{B})$. It can be checked in an obvious way that $\mathbf{A} \times^{\cup} \mathbf{B}$ is nilpotent with degree of nilpotency $\max\{k, l\}$. □

Next we give necessary and sufficient conditions under which the union of two deterministic tree languages is not deterministic.

Theorem 6. *Let S and T be DR ΣX-languages. Then $S \cup T$ is not deterministic if and only if there are a tree $p \in T_\Sigma(X)$, two variables $x, y \in X$ and two different paths $u \in g_x(p)$ and $v \in g_y(p)$ such that $u \in g_x(S)$ and $u \notin g_x(T)$, and $v \in g_y(T)$ and $v \notin g_y(S)$.*

Proof. Assume that $S \cup T$ is not deterministic. Let $\mathbf{A}$ and $\mathbf{B}$ be normalized DR ΣX-recognizers with $S = T(\mathbf{A})$ and $T = T(\mathbf{B})$. Since $S \cup T$ is not deterministic, there is a tree $p \in T_\Sigma(X)$ such that $p \in T(\mathbf{A} \times^{\cup} \mathbf{B})$, $p \notin T(\mathbf{A})$ and $p \notin T(\mathbf{B})$. Therefore, for some $x, y \in X$, $u \in g_x(p)$ and $v \in g_y(p)$ we have $u \in g_x(S) \setminus g_x(T)$, $v \in g_y(T) \setminus g_y(S)$ and $u \neq v$.

Conversely, assume that the conclusions of the theorem hold. Denote by w the maximal initial segment of u and v. Then u and v are of form $u = w\sigma_i u'$ and $v = w\sigma_j v'$, and $i \neq j$. Since $u \in g_x(S)$, there is a $q \in S$ with $u \in g_x(q)$. Similarly, there is a $q' \in T$ with $v \in g_y(q')$. Let r be the tree which is obtained from q' by replacing its subtree at $w\sigma_i$ by the subtree of q at $w\sigma_i$. Obviously, r is not in $S \cup T$, however it is in the closure of $S \cup T$. Therefore, $S \cup T$ is not deterministic. □

Obviously, the trees p satisfying the conditions of the previous theorem are not unary. Therefore, from Theorem 6 we directly obtain

Corollary 7. *Let S and T be two DR ΣX-languages. If $S \setminus T \subseteq T_{\Sigma_1}(X)$ or $T \setminus S \subseteq T_{\Sigma_1}(X)$, then $S \cup T$ is deterministic.* □

This corollary, by Theorem 5, implies

Corollary 8. *Let S and T be two nilpotent DR ΣX-languages. If one of them differs from the other one only in unary trees then $S \cup T$ is nilpotent.* □

Let $p \in T_\Sigma(X) \setminus X$ be a tree. Then $\mathrm{root}(p) = \sigma$ if $p = \sigma(p_1, \ldots, p_m)$. For a tree language $T \subseteq T_\Sigma(X)$, set $\mathrm{root}(T) = \{\mathrm{root}(p) | p \in T \setminus X\}$.

The following result directly follows from Theorems 6 and 5.

Corollary 9. *Let S and T be nilpotent DR ΣX-languages. If*

$$\mathrm{root}(S) \cap (\mathrm{root}(T) = \emptyset,$$

then $S \cup T$ is nilpotent. □

Later we shall use the following obvious result.

Lemma 10. *Let $S, T \subseteq T_\Sigma(X)$ be nilpotent DR tree languages. Then for each $x \in X$, if both $g_x(S)$ and $g_x(T)$ are infinite then $g_x(S) \setminus g_x(T)$ and $g_x(T) \setminus g_x(S)$ are finite.* □

Corollary 11. *Let S and T be nilpotent DR ΣX-languages such that $S \setminus T \not\subseteq T_{\Sigma_1}(X)$. If for an $x \in X$, $g_x(T) \setminus g_x(S)$ is infinite, then $S \cup T$ is not nilpotent.*

Proof. Take a $p \in S \setminus T$ with $p \notin T_{\Sigma_1}(X)$. Then there exist a variable $y \in X$ and a path $u \in g_y(p)$ such that $u \notin g_y(T)$. Assume that the degree of nilpotency of S is k and that of T is l. Then for the variable x satisfying the condition of the corollary, all the trees $r \in T_\Sigma(X)$ of the form $\bar{\text{fr}}(r) = x^t$ with $\text{mh}(p) \geq l$ are in T. Let q be the tree which is obtained from p by replacing each leaf, except for the leaf at u, by a tree r for which $\text{mh}(r) \geq \max\{k, l\}$ and $\bar{\text{fr}}(r) = x^t$ under some t. By Lemma 10, $g_x(S)$ is finite, thus q obviously satisfies the conditions of Theorem 6. Therefore, $S \cup T$ is not nilpotent. □

From Corollary 11 we directly obtain

Corollary 12. *Let S and T be two nilpotent ΣX-languages. If S is finite, T is infinite and $S \setminus T \not\subseteq T_{\Sigma_1}(X)$, then $S \cup T$ is not nilpotent.*

4 Intersection

Let $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ and $\mathbf{B} = (\mathcal{B}, b_0, \beta)$ be DR ΣX-recognizers. Their *intersection direct product* is

$$\mathbf{A} \times^\cap \mathbf{B} = (\mathcal{A} \times \mathcal{B}, (a_0, b_0), \alpha \times^\cap \beta),$$

where $(\alpha \times^\cap \beta)(x) = \alpha(x) \times \beta(x)$ $(x \in X)$.

Theorem 13. *Let $\mathbf{A}$ and $\mathbf{B}$ be DR ΣX-recognizers. Then $T(\mathbf{A}) \cap T(\mathbf{B}) = T(\mathbf{A} \times^\cap \mathbf{B})$.*

Proof. Obvious. □

It is also obvious that the intersection direct product of nilpotent DR tree recognizers is nilpotent. Thus, from Theorem 13, we obtain

Theorem 14. *The class of the nilpotent DR ΣX-languages is closed under intersection.*

5 Complementation

Let $T \subseteq T_\Sigma(X)$ be a tree language. The complement $T_\Sigma(X) \setminus T$ of T will be denoted by $\text{c}(T)$. Moreover, for all tree languages $T \subseteq T_\Sigma(X)$ and variables $x \in X$, $T(x)$ will stand for $T \cap T_{\Sigma_1}(\{x\})$, i.e. $T(x)$ consists of all (unary) trees from T whose leaves are x.

Lemma 15. *Assume that $\Sigma_i = \emptyset$ for all $i > 1$. Then a tree language $T \subseteq T_\Sigma(X)$ is nilpotent if and only if $T(x)$ or $\mathrm{c}(T)(x)$ is finite for each $x \in X$.*

Proof. If T is nilpotent, then obviously, each $T(x)$ $(x \in X)$ is nilpotent. Since in the unary case we can apply the well known characterization of nilpotent string languages, $T(x)$ or $\mathrm{c}(T)(x)$ is finite.

Conversely, assume that the conclusions of our lemma hold. Observe that in this special case the path language $g_x(S)$ of S and the path language $g_x(S(x))$ of $S(x)$ coincide for all $x \in X$ and $S \subseteq T_\Sigma(X)$. Moreover, again by a well known classical characterization of nilpotent string languages (see, [6]), the syntactic semigroups of nilpotent string languages are nilpotent. Therefore, the syntactic semigroup of $g_x(T)$ and that of $g_x(\mathrm{c}(T))$ are nilpotent. This, by the proof of Theorem 5 in [3], implies that both T and $\mathrm{c}(T)$ are nilpotent. □

From the above theorem we directly obtain

Corollary 16. *If $\Sigma_i = \emptyset$ for all $i > 1$, then a tree language $T \subseteq T_\Sigma(X)$ is nilpotent if and only if its complement $\mathrm{c}(T)$ is nilpotent.* □

Lemma 17. *Suppose that Σ contains at least one operational symbol with arity greater than 1, and let $T \subseteq T_{\Sigma_1}(X)$ be a tree language. Then T is nilpotent if and only if it is finite. Moreover, if T is nilpotent then so is its complement $\mathrm{c}(T)$.*

Proof. Assume that T is infinite and nilpotent, and that it is recognized by the nilpotent DR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ with degree of nilpotency k. Let $\bar{a}$ be the nilpotent element of $\mathbf{A}$. Since T is infinite, there exists a $p(x) \in T$ with $\mathrm{h}(p) \geq k$. Therefore, $\bar{a} \in \alpha(x)$. Thus, all trees $q \in T_\Sigma(\{x\})$ with $\mathrm{mh}(q) \geq k$ are also in T, which contradicts the assumption that $T \subseteq T_{\Sigma_1}(X)$.

Conversely, assume that $T \subseteq T_{\Sigma_1}(X)$ is finite. Construct a DR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ in the following way. Let $k = \max\{\mathrm{h}(p) \mid p \in T\}$. Set

$$A = \{u \in \hat{\Sigma}^* \mid |u| \leq k\} \cup \{*\}.$$

Moreover, for all $m > 0$, $\sigma \in \Sigma_m$ and $u \in \hat{\Sigma}^*$, let

$$\sigma^{\mathcal{A}}(u) = (u\sigma_1, \ldots, u\sigma_m),$$

if $|u| < k$, and

$$\sigma^{\mathcal{A}}(u) = \sigma^{\mathcal{A}}(*) = (*, \ldots, *),$$

otherwise. Finally, let $a_0 = e$, and $\alpha(x) = g_x(T)$ $(x \in X)$. It is obvious that $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ is nilpotent and $T = T(\mathbf{A})$. It is also clear that $\mathbf{A}' = (\mathcal{A}, a_0, \alpha')$ with $\alpha'(x) = A \setminus \alpha(x)$ $(x \in X)$ recognizes $\mathrm{c}(T)$. □

Lemma 18. *Suppose that Σ contains at least one operational symbol with arity greater than 1, and let $T \subseteq T_\Sigma(X)$ be an infinite nilpotent tree language. If $\mathrm{c}(T) \not\subseteq T_{\Sigma_1}(X)$, then $\mathrm{c}(T)$ is not nilpotent.*

Proof. Suppose that T can be recognized by a nilpotent DR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ with degree of nilpotency k. Let $\bar{a}$ be the nilpotent element of $\mathbf{A}$. Since T is infinite, there is a $\bar{z} \in X$ such that $\bar{a} \in \alpha(\bar{z})$. Assume that $\mathrm{c}(T) \not\subseteq T_{\Sigma_1}(X)$ is nilpotent and can be recognized by the nilpotent DR ΣX-recognizer $\mathbf{B} = (\mathcal{B}, b_0, \beta)$ with the nilpotent element $\bar{b}$. Suppose that the degree of nilpotency of $\mathbf{B}$ l. Take a tree $p \in \mathrm{c}(T)$ with $p \notin T_{\Sigma_1}(X)$. Then for some (not necessarily different) variables $x, y \in X$, there are different paths $u \in g_x(p)$ and $v \in g_y(p)$ such that $a_0 u \notin \alpha(x)$ or $a_0 v \notin \alpha(y)$. Assume that $a_0 u \notin \alpha(x)$. Suppose that $l \geq k$. Replace in p the variable y at v with an arbitrary $r \in T_\Sigma(\{z\})$ $(z \in X)$ of height greater than or equal to l, and denote the resultant tree by q. Obviously, $q \in \mathrm{c}(T)$. Then $\bar{b} \in \beta(z)$ for all $z \in X$. Thus, for every tree $t \in T_\Sigma(X)$ with $\mathrm{mh}(t) \geq l$ we have $t \in \mathrm{c}(T)$. Moreover, by our assumptions, every tree $t \in T_\Sigma(\{\bar{z}\})$ with $\mathrm{mh}(t) \geq l$ is also in T, which is a contradiction. The case $k > l$ can be treated in a similar way. □

Using Lemma 18, we give a simple example showing that there exists a nilpotent tree language whose complement is not nilpotent.

Example 19. Let $\Sigma = \sigma_2 = \{\sigma\}$ and $X = \{x, y\}$. Take the DR Σ-algebra $\mathcal{A} = (A, \Sigma)$ with $A = \{a_0\}$ and $\sigma^{\mathcal{A}}(a_0) = (a_0, a_0)$. Finally, let $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ be the ΣX-recognizer, where $\alpha(x) = \{a_0\}$ and $\alpha(y) = \emptyset$. Obviously, $\mathbf{A}$ is nilpotent and $T(\mathbf{A}) = T_\Sigma(\{x\})$. Since $T(\mathbf{A})$ is infinite and $\Sigma_1 = \emptyset$, by Lemma 18, $\mathrm{c}(T(\mathbf{A}))$ is not nilpotent.

We now state a result characterizing those DR tree languages T for which both T and $\mathrm{c}(T)$ are nilpotent. The case $\Sigma = \Sigma_1$ is settled by Lemma 15.

Theorem 20. *Suppose that Σ contains at least one operational symbol with arity greater than 1, and let $T \subseteq T_\Sigma(X)$ be a tree language. Then T and $\mathrm{c}(T)$ are simultaneously nilpotent if and only if one of the following two statements is true:*

(i) *$T \subseteq T_{\Sigma_1}(X)$ and T is finite.*

(ii) *$\mathrm{c}(T) \subseteq T_{\Sigma_1}(X)$ and $\mathrm{c}(T)$ is finite.*

Proof. If (i) holds, then, by Lemma 17, both T and $\mathrm{c}(T)$ are nilpotent. Case (ii) can be treated in the same way.

Conversely, assume that T and $\mathrm{c}(T)$ are simultaneously nilpotent. If T is finite, then $\mathrm{c}(T)$ is infinite. Thus, by Lemma 18, $T \subseteq T_{\Sigma_1}(X)$. Therefore (i) holds. If T is infinite, then $\mathrm{c}(T) \subseteq T_{\Sigma_1}(X)$ by Lemma 18. From this, using the assumption that $\mathrm{c}(T)$ is nilpotent, by Lemma 17, we obtain that $\mathrm{c}(T)$ is finite. Therefore, (ii) holds. □

References

[1] Courcelle, B.: A representation of trees by languages I. *Theoretical Computer Science* **6** (1978), 255-279.

[2] Gécseg, F. and Imreh, B.: On a special class of tree automata, *2nd Conf. on Automata, Languages and Programming Systems,* Salgótarján, Hungary 1988. 141-152.

[3] Gécseg, F. and Imreh, B.: On definite and nilpotent DR tree languages, *Journal of Automata, Languages and Combinatorics,* **9:1** (2004), 55-60.

[4] Gécseg, F. and Steinby, M.: Minimal Recognizers and Syntactic Monoids of DR Tree Languages, in Words, Semigroups, & Transductions, World Scientifics (2001), 155-167.

[5] Jurvanen, E.: The Boolean closure of DR-recognizable tree languages, *Acta Cybernetica,* **10** (1992), 255-272.

[6] Ševrin, L. N.: On some classes of abstract automata. *Uspehi matem. nauk,* **17:6 (108)** (1962), 219.

[7] Virágh, J.: Deterministic ascending tree automata I. *Acta Cybernetica* **5** (1980), 33-42.

Received April, 2005

Acta Cybernetica **17** (2006) 459–470.

Finitely Presentable Tree Series

Symeon Bozapalidis* and Olympia Louskou-Bozapalidou†

Abstract

Tree height is known to be a non-recognizable series. In this paper, we detect two remarkable classes where this series belongs: that of polynomially presentable tree series and that of almost linearly presentable tree series.

Both the above classes have nice closure properties, and seem to constitute the first levels of a tree series hierarchy which starts from the class of recognizable treeseries.

1 Introduction

It is well known that some tree functions of wide use in computer science fail to be recognizable, that is they can not be obtained as behaviors of tree automata weighted over a certain semiring. Berstel and Reutenauer proved that the tree series *height* : $T_\Gamma \to \mathbb{N}$ sending every tree t over the ranked alphabet Γ to its height is non-recognizable (cf. [BR]). Therefore it is quite natural to search for classes having good closure properties in which this tree series belongs.

In this paper, we give two such classes: the class PP of polynomially presentable tree series and the class ALP of almost linearly presentable tree series.

Both PP and ALP are closed under sum, scalar product, top-catenation, left derivative and semiring morphism.

Given a finite ranked alphabet Γ and a semiring K we denote by $K\langle\langle T_\Gamma \rangle\rangle$ the set of all tree series $S : T_\Gamma \to K$, equipped with the standard operations of sum, scalar product and top-catenation.

We say that a tree series $S : T_\Gamma \to K$ is *polynomially presentable* whenever it belongs to a finitely generated invariant subalgebra of $K\langle\langle T_\Gamma \rangle\rangle$. Also, $S : T_\Gamma \to K$ is said to be *linearly presentable* whenever it belongs to a finitely generated invariant K-subsemimodule of $K\langle\langle T_\Gamma \rangle\rangle$.

The reader is assumed to be familiar with semirings, semimodules etc (for details, see [SS], [KS]).

*Department of Mathematics, Aristotle University of Thessaloniki, GR-54006, Thessaloniki, Greece.

†Technical Institute of Western Macedonia, Kozani, Greece.

2 Basic Facts

2.1 Trees

In this subsection we briefly exhibit the tree substitution operations used throughout this paper.

Given a finite ranked alphabet $\Gamma = (\Gamma_k)_{k\geq 0}$ and a set of variables $X_n = \{x_1, \ldots, x_n\}$, we denote by $T_\Gamma(X_n)$ the smallest set verifying next two items:

- $\Gamma_0 \cup X_n \subseteq T_\Gamma(X_n)$ and
- for $f \in \Gamma_k$, $k \geq 1$, and $t_1, \ldots, t_k \in T_\Gamma(X_n)$ the word $f(t_1, \ldots, t_k) \in T_\Gamma(X_n)$.

For $n = 0$, $T_\Gamma(X_n)$ is written as T_Γ. The elements of $T_\Gamma(X_n)$ are called trees over Γ indexed by the variables $x_1, \ldots, x_n$.

The *height* of a tree $t \in T_\Gamma$, denoted by $height(t)$ is inductively defined by

- $height(c) = 0$, for all $c \in \Gamma_0$ and
- $height(f(t_1, \ldots, t_n)) = 1 + \max\{height(t_i) \mid 1 \leq i \leq n\}$.

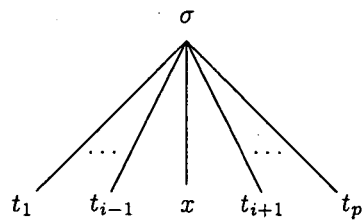
Consider trees

$$t \in T_\Gamma(X_n), t_1, \ldots, t_n, t_1^{(i)}, \ldots, t_{\lambda_i}^{(i)} \in T_\Gamma(X_n), 1 \leq i \leq n$$

where we assume that the variable x_i occurs exactly $\lambda_i \geq 0$ times in the tree t. We use the notation:

- $t[t_1/x_1, \ldots, t_n/x_n]$ or simply $t[t_1, \ldots, t_n]$ for the result of substituting t_i for every occurrence of x_i in t.
- $t\left[\left(t_1^{(1)}, \ldots, t_{\lambda_1}^{(1)}\right)/x_1, \ldots, \left(t_1^{(n)}, \ldots, t_{\lambda_n}^{(n)}\right)/x_n\right]$ for the result of substituting $t_1^{(i)}, \ldots, t_{\lambda_i}^{(i)}$ for the occurrences of x_i in t from left to right $(1 \leq i \leq n)$.

Consider now the subset P_Γ of $T_\Gamma(x)$ consisting of all trees where the variable x occurs once. P_Γ becomes a monoid, with multiplication the substitution at x; precisely, if $\tau, \pi \in P_\Gamma$, $\tau\pi$ is the tree obtained by substituting π for x in τ. Actually, P_Γ is the free monoid generated by the trees of the following form:



$\sigma \in \Gamma_p$, $p \geq 1$, $t_j \in T_\Gamma$, $(j \neq i)$.

Figure 1:

On other hand, P_Γ acts canonically on T_Γ :

$$P_\Gamma \times T_\Gamma \to T_\Gamma \qquad (\tau, t) \mapsto \tau t = \tau\,[t/x]\,.$$

For $\tau \in P_\Gamma$, $|\tau|$ denotes its length in the free monoid P_Γ. If τ is as in Figure 1 then $|\tau| = 1$ while if $\tau = \tau_1 \cdot \tau_2$, then $|\tau| = |\tau_1| + |\tau_2|$.

2.2 Formal Series on Trees

Assume a ranked alphabet Γ and a set of variables $X_n = \{x_1, \ldots, x_n\}$ are given, as well as a semiring K.

The functions $S : T_\Gamma(X_n) \to K$ are called *tree series.*

The value of S at $t \in T_\Gamma(X_n)$ is denoted by (S, t) and is refered to as the *coefficient* of S at t.

The set $K\langle\langle T_\Gamma(X_n)\rangle\rangle$ of tree series on $T_\Gamma(X_n)$ is converted into a K-semimodule when addition and scalar multiplication are point wisely defined:

$$\begin{aligned}(S_1 + S_2, t) &= (S_1, t) + (S_2, t)\\ (\lambda S, t) &= \lambda\,(S, t)\end{aligned}$$

for all $t \in T_\Gamma(X_n)$, $\lambda \in K$ and $S_1, S_2, S \in K\langle\langle T_\Gamma(X_n)\rangle\rangle$.

Moreover a partial infinite addition on $K\langle\langle T_\Gamma(X_n)\rangle\rangle$ can be defined as follows: we say that a family of tree series $(S_i)_{i\in I}$ is *locally finite* whenever for each $t \in T_\Gamma(X_n)$ the set $\{i \mid (S_i, t) \neq 0\}$ is finite. Then $\sum_{i\in I} S_i$ exists and is given by

$$\left(\sum_{i\in I} S_i, t\right) = \sum_{i\in I} (S_i, t) \qquad \text{for all } t \in T_\Gamma(X_n)\,.$$

According to this discussion every $S \in K\langle\langle T_\Gamma(X_n)\rangle\rangle$ can be represented as an infinite sum

$$S = \sum_{t\in T_\Gamma(X_n)} (S, t)\, t.$$

The support of a series $S : T_\Gamma(X_n) \to K$ is the tree language

$$supp\,(S) = \{t \in T_\Gamma(X_n) \mid (S, t) \neq 0\}\,.$$

Series $S \in K\langle\langle T_\Gamma(X_n)\rangle\rangle$ whose support is finite are termed *polynomials* and their set is denoted by $K\langle T_\Gamma(X_n)\rangle$.

Given $\sigma \in \Gamma_p$ and $S_1, \ldots, S_p \in K\langle\langle T_\Gamma(X_n)\rangle\rangle$, the *$\sigma$-top catenation series*

$$\sigma\,(S_1, \ldots, S_p) : T_\Gamma(X_n) \to K$$

is defined as follows. For $t \in T_\Gamma(X_n)$

$$(\sigma\,(S_1, \ldots, S_p), t) = (S_1, t_1) \cdots (S_p, t_p) \text{ if } t = \sigma\,(t_1, \ldots, t_p) \text{ and } 0 \text{ else.}$$

More generally, for every $n \geq 0$, $t \in T_\Gamma(X_n)$ and $S_1, \ldots, S_n \in K\langle\langle T_\Gamma \rangle\rangle$ we define the series

$$t[S_1, \ldots, S_n] : T_\Gamma(X_n) \to K$$

inductively by the clauses

- $x_i[S_1, \ldots, S_n] = S_i, \qquad 1 \leq i \leq n$
- $c[S_1, \ldots, S_n] = c, \qquad c \in \Gamma_0$
- $\sigma(t_1, \ldots, t_p)[S_1, \ldots, S_n] = \sigma(t_1[S_1, \ldots, S_n], \ldots, t_p[S_1, \ldots, S_n])$, for $\sigma \in \Gamma_p, t_j \in T_\Gamma(X_n)$.

Proposition 1. *For every $n \geq 1, t \in T_\Gamma(X_n), S_1, \ldots, S_n \in K\langle\langle T_\Gamma \rangle\rangle$ and $s \in T_\Gamma$,*

$$(t[S_1, \ldots, S_n], s) = \left(S_1, t_1^{(1)}\right) \cdots \left(S_1, t_{\lambda_1}^{(1)}\right) \cdots \left(S_n, t_1^{(n)}\right) \cdots \left(S_n, t_{\lambda_n}^{(n)}\right)$$

if there are $t_1^{(i)}, \ldots, t_{\lambda_i}^{(i)} \in T_\Gamma$, $1 \leq i \leq n$ such that

$$s = t\left[\left(t_1^{(1)}, \ldots, t_{\lambda_1}^{(1)}\right)/x_1, \ldots, \left(t_1^{(n)}, \ldots, t_{\lambda_n}^{(n)}\right)/x_n\right].$$

and $(t[S_1 \ldots, S_n], s) = 0$, otherewise.

By linear extension, we can define $p[S_1, \ldots, S_n]$ for any polynomial $p \in K\langle T_\Gamma(X_n)\rangle$

$$p[S_1, \ldots, S_n] = \sum_{t \in T_\Gamma(X_n)} (p, t)\, t[S_1, \ldots, S_n].$$

The last operation we need is derivation. The derivative of $S \in K\langle\langle T_\Gamma \rangle\rangle$ at $\tau \in P_\Gamma$ is a tree series

$$\tau^{-1}S = \sum_{t \in T_\Gamma} (S, \tau t)\, t.$$

The derivation has the following properties:

1. $\tau^{-1}\left(\pi^{-1}S\right) = (\pi\tau)^{-1}S$, for all $\tau, \pi \in P_\Gamma, S \in K\langle\langle T_\Gamma \rangle\rangle$,
2. $\tau^{-1}(\sigma(S_1, \ldots, S_p)) = \prod_{j \neq i} (S_j, t_j)\, \pi^{-1}S_i$, if $\tau = \sigma(t_1, \ldots, t_{i-1}, \pi, t_{i+1}, \ldots, t_p)$, for every $\tau \in P_\Gamma$ and $S_1, \ldots, S_p \in K\langle\langle T_\Gamma \rangle\rangle$,
3. for every $\tau \in P_\Gamma$, index set I and family $(S_i, i \in I)$ over $K\langle\langle T_\Gamma \rangle\rangle$, if $\sum_{i \in I} S_i$ exists, then $\sum_{i \in I} \tau^{-1}S_i$ also exists and $\tau^{-1}\left(\sum_{i \in I} S_i\right) = \sum_{i \in I} \tau^{-1}S_i$.

2.3 Recognizable Tree series

Recall that a K-Γ-tree automaton is a triple $\mathcal{M} = (Q, \mu, T)$ consisting of a finite set Q of states, a final state function $T : Q \to K$ and a Γ-indexed family of functions

$$\mu = \left(\mu_f : Q^n \to K^Q\right)_{f \in \Gamma_n, n \geq 0}$$

describing the moves of $\mathcal{M}$.

The function $\mu_f : Q^n \to K^Q$ is multilinearly extended into a function

$$\bar{\mu}_f : \left(K^Q\right)^n \to K^Q$$

by the formula

$$\bar{\mu}_f(X_1, \ldots, X_n) = \sum_{q_1, \ldots, q_n \in Q} X_1(q_1) \cdots X_n(q_n)\, \mu_f(q_1, \ldots, q_n).$$

Then the behaviour of $\mathcal{M}$ is the series $|\mathcal{M}| : T_\Gamma \to K$ defined by

$$(|\mathcal{M}|, t) = \sum_{q \in Q} \mu_{\mathcal{M}}(t)(q) \cdot T(q)$$

where $\mu_{\mathcal{M}} : T_\Gamma \to K^Q$ is inductively given by the clause

$$\mu_{\mathcal{M}}(f(t_1, \ldots, t_n)) = \bar{\mu}_f(\mu_{\mathcal{M}}(t_1), \ldots, \mu_{\mathcal{M}}(t_n)),\ f \in \Gamma_n, n \geq 0, t_1, \ldots, t_n \in T_\Gamma.$$

A tree series $S : T_\Gamma \to K$ is called *recognizable* whenever it is the behaviour of a K-Γ-tree automaton. $REC(K,\Gamma)$ stands for the so obtained class.

The tree series $S : T_\Gamma \to \mathbb{N}$ sending every tree $t \in T_\Gamma$ to its size (i.e. the number of symbols of Γ occurring in t), is recognizable. On the contrary, the tree series *height* $: T_\Gamma \to \mathbb{N}$ fails to be recognizable (cf. [BR]).

3 Subalgebras of $K\langle\langle T_\Gamma \rangle\rangle$

A subset $\mathcal{A} \subseteq K\langle\langle T_\Gamma \rangle\rangle$ closed under sum, scalar product and σ-top catenation (for all $\sigma \in \Gamma_p, p \geq 1$) is termed a subalgebra of $K\langle\langle T_\Gamma \rangle\rangle$.

Proposition 2. *$\mathcal{A} \subseteq K\langle\langle T_\Gamma \rangle\rangle$ is a subalgebra iff for each polynomial $p \in K\langle T_\Gamma(X_n)\rangle$ and a sequence of series $S_1, \ldots, S_n \in \mathcal{A}, p[S_1, \ldots, S_n] \in \mathcal{A}$.*

The intersection of any family of subalgebras of $K\langle\langle T_\Gamma \rangle\rangle$ is again a subalgebra and thus we can speak of the subalgebra generated by a subset $\mathcal{S} \subseteq K\langle\langle T_\Gamma \rangle\rangle$. It is denoted by $\langle\mathcal{S}\rangle_{K,\Gamma}$.

Proposition 3. *For every $\mathcal{S} \subseteq K\langle\langle T_\Gamma \rangle\rangle$, we have*

$$\langle\mathcal{S}\rangle_{K,\Gamma} = \{p[S_1, \ldots, S_n] \mid p \in K\langle T_\Gamma(X_n)\rangle, S_1, \ldots, S_n \in \mathcal{S}, n \geq 0\}.$$

Proof. Let

$$U = \{p[S_1, \ldots, S_n] \mid p \in K\langle T_\Gamma(X_n)\rangle, S_1, \ldots, S_n \in \mathcal{S}, n \geq 0\}.$$

Certainly $\mathcal{S} \subseteq U$. Next we show that U is a subalgebra of $K\langle\langle T_\Gamma\rangle\rangle$, i.e., that for every $n \geq 0, p \in K\langle T_\Gamma(X_n)\rangle$ and $p_i \in K\langle T_\Gamma(X_{k_i})\rangle, S_1^{(i)}, \ldots, S_{k_i}^{(i)} \in \mathcal{S}$ $1 \leq i \leq n$ it holds

$$p\left[p_1\left[S_1^{(1)}, \ldots, S_{k_1}^{(1)}\right], \ldots, p_n\left[S_1^{(n)}, \ldots, S_{k_n}^{(n)}\right]\right] \in U.$$

We introduce the polynomial $\bar{p}_i, 1 \leq i \leq n$ by setting

$$\bar{p}_1 = p_1,$$

$$\bar{p}_2 = p_2[x_{k_1+1}/x_1, \ldots, x_{k_1+k_2}/x_{k_2}],$$

$$\vdots$$

$$\bar{p}_n = p_n\left[x_{k_1+\cdots+k_{n-1}+1}/x_1, \ldots, x_{k_1+\cdots+k_{n-1}+k_n}/x_{k_n}\right].$$

Then

$$p\left[p_1\left[S_1^{(1)}, \ldots, S_{k_1}^{(1)}\right], \ldots, p_n\left[S_1^{(n)}, \ldots, S_{k_n}^{(n)}\right]\right] =$$

$$p[\bar{p}_1, \ldots, \bar{p}_n]\left[S_1^{(1)}/x_1, \ldots, S_{k_1}^{(1)}/x_{k_1}, S_1^{(2)}/x_{k_1+1}, \ldots,\right.$$

$$\left.S_{k_2}^{(2)}/x_{k_1+k_2}, \ldots, S_1^{(n)}/x_{k_1+\cdots+k_{n-1}+1}, \ldots, S_{k_n}^{(n)}/x_{k_1+\cdots+k_n}\right].$$

Since $p[\bar{p}_1, \ldots, \bar{p}_n] \in K\langle\langle T_\Gamma(X_{k_1+\cdots+k_n})\rangle\rangle$ the result comes by aplying Proposition 2.

Now, let $\overline{U}$ be a subalgebra of $K\langle\langle T_\Gamma\rangle\rangle$ including $\mathcal{S}$. Then for any $p[S_1, \ldots, S_n] \in U$ with $p \in K\langle T_\Gamma(X_n)\rangle$ and $S_1, \ldots, S_n \in \mathcal{S}$, we have $p[S_1, \ldots, S_n] \in \overline{U}$ and thus U is the smallest subalgebra of $K\langle\langle T_\Gamma\rangle\rangle$ including $\mathcal{S}$, i.e. $U = \langle\mathcal{S}\rangle_{K,\Gamma}$. □

A subalgebra $\mathcal{A} \subseteq K\langle\langle T_\Gamma\rangle\rangle$ is said to be *invariant* if it is closed under derivation, i.e.

$$S \in \mathcal{A} \text{ and } \tau \in P_\Gamma \text{ implies } \tau^{-1}S \in \mathcal{A}.$$

Proposition 4. *The subalgebra $\langle\mathcal{S}\rangle$ generated by $\mathcal{S} \subseteq K\langle T_\Gamma\rangle$ is invariant iff it contains the derivatives of all its generators*

Proof. One direction is obvious.

To establish the opposite direction we first show that if $\sigma \in \Gamma_k$ and $S_1, \ldots, S_k \in \mathcal{S}$ then

$$\tau^{-1}\sigma(S_1, \ldots, S_k) \in \langle\mathcal{S}\rangle, \quad \text{for all } \tau \in P_\Gamma.$$

Indeed, if $\tau = \sigma(t_1, \ldots, t_{i-1}, \pi, t_{i+1}, \ldots, t_k)$ then for all $t \in T_\Gamma$

$$\begin{aligned}(\tau^{-1}\sigma(S_1, \ldots, S_k), t) &= (\sigma(S_1, \ldots, S_k), \tau t)\\ &= \prod_{j \neq i}(S_j, t_j)(S_i, \pi t)\\ &= \alpha(\pi^{-1}S_i, t),\end{aligned}$$

where $\alpha = \prod_{j \neq i}(S_j, t_j)$. In other words

$$\tau^{-1}\sigma(S_1, \ldots, S_k) = \alpha\pi^{-1}S_i \in \langle \mathcal{S} \rangle\,, \quad \alpha \in K$$

since, by hypothesis, $\langle \mathcal{S} \rangle$ contains all the derivatives of its generators.

In all other instances of τ, it holds

$$\tau^{-1}\sigma(S_1, \ldots, S_k) = 0 \in \langle \mathcal{S} \rangle\,.$$

By induction on the complexity of $t \in T_\Gamma(X_n)$ we show that $\tau^{-1}t[S_1, \ldots, S_n] \in \langle \mathcal{S} \rangle$.

For $t \in \Gamma_0 \cup X_n$ we have nothing to prove. Let $t = \sigma(t_1, \ldots, t_k)$; then

$$\begin{aligned}\tau^{-1}t[S_1, \ldots, S_n] &= \tau^{-1}\sigma(t_1, \ldots, t_k)[S_1, \ldots, S_n] \\ &= \tau^{-1}\sigma(t_1[S_1, \ldots, S_n], \ldots, t_k[S_1, \ldots, S_n]) \in \langle \mathcal{S} \rangle\end{aligned}$$

Furthermore, for any polynomial $p \in K\langle T_\Gamma(X_n) \rangle$ we have

$$\tau^{-1}p[S_1, \ldots, S_n] = \Sigma_{t \in T_\Gamma(X_n)}(p, t)\tau^{-1}t[S_1, \ldots, S_n] \in \langle \mathcal{S} \rangle$$

where the above sum is finite.

□

4 Finitely Presentable Tree series

A series $S \in K\langle\langle T_\Gamma \rangle\rangle$ is said to be *linearly presentable* if there exist series $S_1, \ldots, S_n \in K\langle\langle T_\Gamma \rangle\rangle$ with the following two properties

1. There are $\lambda_1, \ldots, \lambda_n \in K$ such that S is expressed as a linear combination of them
$$S = \lambda_1 S_1 + \cdots + \lambda_n S_n,\ \lambda_i \in K$$
and
2. for each index i $(1 \leq i \leq k)$ and each $\tau \in P_\Gamma$, there are $\mu_{i_1}, \ldots, \mu_{i_k} \in K$ such that for each index i $(1 \leq i \leq n)$ and each $\tau \in P_\Gamma$
$$\tau^{-1}S_i = \sum_{i=1}^{n} \mu_{ij} S_j,\ \mu_{ij} \in K, 1 \leq i \leq n.$$

We denote by $LP(K, \Gamma)$ the class of linearly presentable tree series.

Proposition 5. $REC(\Gamma, K) \subseteq LP(\Gamma, K)$.

Proof. Consider a K-Γ-tree automaton $\mathcal{M} = (Q, \mu, T)$, its associated system

$$x_q = \sum_{\substack{k \geq 0, f \in \Gamma_k \\ q_1, \ldots, q_k \in Q}} \mu_f(q_1, \ldots, q_k)(q)\, f(x_{q_1}, \ldots, x_{q_k}) \qquad (\Sigma M)$$

and for all $q \in Q$, the K-Γ-tree automaton $\mathcal{M}_q = (Q, \mu, \hat{q})$ with $\hat{q} : Q \to K$ defined by $\hat{q}(p) = 1$, if $p = q$ and $\hat{q}(p) = 0$, else.

It is known [Bo2] that the tuple $(|\mathcal{M}_q|)_{q \in Q}$ is the unique solution of (ΣM)

$$|\mathcal{M}_q| = \sum_{\substack{f \in \Gamma_k, k \geq 0 \\ q_1, \ldots, q_k \in Q}} \mu_f(q_1, \ldots, q_k)(q)\, f(|\mathcal{M}_{q_1}|, \ldots, |\mathcal{M}_{q_k}|). \qquad (\star)$$

By construction we have

$$|\mathcal{M}| = \sum_{q \in Q} T(q)\, |\mathcal{M}_q|$$

that is $|\mathcal{M}|$ is linear combination of the series $|\mathcal{M}_q|, q \in Q$. The proof will be completed if for each tree $\tau \in P_\Gamma$ of the form $\tau = g(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_k), g \in \Gamma_k, t \in T_\Gamma$ and for each state $q \in Q$ show that $\tau^{-1}|\mathcal{M}_q|$ can also be written as linear combination of $|\mathcal{M}_q|, q \in Q$.

Derivating $(\star)$ at τ we get

$$\begin{aligned}
(\tau^{-1}|\mathcal{M}_q|, s) &= (|\mathcal{M}_q|, \tau s) \\
&= \sum_{\substack{f \in \Gamma_k, k \geq 0 \\ q_1, \ldots, q_k \in Q}} \mu_f(q_1, \ldots, q_k)(q)\, (f(|\mathcal{M}_{q_1}|, \ldots, |\mathcal{M}_{q_k}|), \tau s) \\
&= \sum_{q_1, \ldots, q_k \in Q} \mu_g(q_1, \ldots, q_k)(q)\, (g(|\mathcal{M}_{q_1}|, \ldots, |\mathcal{M}_{q_k}|), \tau s) \\
&= \sum_{q_1, \ldots, q_k \in Q} \mu_g(q_1, \ldots, q_k)(q)\, (|\mathcal{M}_{q_1}|, t_1) \cdots (|\mathcal{M}_{q_{i-1}}|, t_{i-1}) \\
&\qquad (|\mathcal{M}_{q_i}|, s)\, (|\mathcal{M}_{q_{i+1}}|, t_{i+1}) \cdots (|\mathcal{M}_{q_k}|, t_k) \\
&= \sum_{q_i \in Q} \lambda_{q_i, \tau}\, (|\mathcal{M}_{q_i}|, s).
\end{aligned}$$

In other words

$$\tau^{-1}|\mathcal{M}_q| = \sum_{q_i \in Q} \lambda_{q_i, \tau}\, |\mathcal{M}_{q_i}|$$

as wanted. □

A tree series $S \in K\langle\langle T_\Gamma \rangle\rangle$ is said to be *polynomially presentable* if there is a finite subset $\mathcal{S} \subseteq K\langle\langle T_\Gamma \rangle\rangle$ satisfying the following two conditions:

1. there is a polynomial $p \in K\langle T_\Gamma(X_n)\rangle$ and there are $S_1, \ldots, S_n \in \mathcal{S}$ such that $S = p[S_1, \ldots, S_n]$ and

2. for every $\tau \in P_\Gamma$ and $S \in \mathcal{S}$, there is a polynomial $p_{\tau,S} \in K\langle T_\Gamma(X_n)\rangle$ and $S_1, \ldots, S_n \in \mathcal{S}$ such that $\tau^{-1}(S) = p_{\tau,S}[S_1, \ldots, S_n]$.

Let us denote the class of polynomially presentable tree series by $PP(K,\Gamma)$. It should be clear that linearly presentable tree series are also polynomially presentable, hence $LP(\Gamma,K) \subseteq PP(\Gamma,K)$.

Moreover, by Proposition 2 and the definition of an invariant subalgebra, S is polynomially presentable if and only if it is an element of an invariant, finitely generated subalgebra of $K\langle\langle T_\Gamma\rangle\rangle$.

Closure properties of polynomially presentable tree series are examined below.

Proposition 6. *The family $PP(K,\Gamma)$ of polynomially presentable series of $K\langle\langle T_\Gamma\rangle\rangle$ is an invariant subalgebra of $K\langle\langle T_\Gamma\rangle\rangle$. Moreover if $\phi : K \to \Lambda$ is a semiring morphism and $S \in PP(K,\Gamma)$, then $S \circ \phi \in PP(\Gamma,\Lambda)$, where $(S \circ \phi, t) = \phi(S(t))$, for all $t \in T_\Gamma$.*

Proof. According to Proposition 4, if $\mathcal{A}$ and $\mathcal{A}'$ are invariant subalgebras of $K\langle\langle T_\Gamma\rangle\rangle$ generated by the lists $T_1,\ldots,T_k$ and $T'_1,\ldots,T'_\lambda$ respectively, then the subalgebra generated by the joint list $T_1,\ldots,T_k,T'_1,\ldots T'_\lambda$ is automatically invariant. In other words we may assume that any finite set of finitely presentable series is included into the same invariant finitely generated subalgebra.

Thus, if $S_1,\ldots,S_n \in PP(K,\Gamma)$ and $p \in K\langle T_\Gamma\rangle$, then there exist series $T_1,\ldots,T_k$ so that

$$S_i = p_i[T_1,\ldots,T_k],\ p_i \in K\langle T_\Gamma(X_k)\rangle, i = 1,\ldots,n$$

and for all $\tau \in P_\Gamma$

$$\tau^{-1}T_j = p_{j,\tau}[T_1,\ldots,T_k],\ p_{j,\tau} \in K\langle T_\Gamma(X_k)\rangle, j = 1,\ldots,k.$$

We have

$$p[S_1,\ldots,S_n] = p[p_1[T_1,\ldots,T_k],\ldots,p_n[T_1,\ldots,T_k]] = p[p_1,\ldots,p_n][T_1,\ldots,T_k].$$

Since $p[p_1,\ldots,p_n]$ is polynomial, we get

$$p[S_1,\ldots,S_n] \in PP(K,\Gamma).$$

Therefore, by virtue of Proposition 2, $PP(K,\Gamma)$ is a subalgebra of $K\langle\langle T_\Gamma\rangle\rangle$.

Next we establish the following identities

$$\tau^{-1}(\phi \circ S) = \phi \circ (\tau^{-1}S), \quad \phi \circ (p[S_1,\ldots,S_n]) = (\phi \circ p)[\phi \circ S_1,\ldots,\phi \circ S_n]$$

holding for all $\tau \in P_\Gamma, S, S_1,\ldots,S_n \in K\langle\langle T_\Gamma\rangle\rangle, p \in K\langle T_\Gamma(X_n)\rangle$ and any semiring morphism $\phi : K \to \Lambda$.

Indeed for all $s \in T_\Gamma$ we have

$$(\tau^{-1}(\phi \circ S), s) = (\phi \circ S, \tau s) = \phi(S, \tau s) = \phi(\tau^{-1}S, s) = (\phi \circ (\tau^{-1}S), s)$$

and

$$(\phi \circ (p[S_1,\ldots,S_n]),s) = \phi(p[S_1,\ldots,S_n],s)$$
$$= \phi(p,t)\left(S_1,s_1^{(1)}\right)\cdots\left(S_1,s_{k_1}^{(1)}\right)\cdots\left(S_1,s_1^{(n)}\right)\cdots\left(S_1,s_{k_n}^{(n)}\right)$$
$$= (\phi\circ p,t)\left(\phi\circ S_1,s_1^{(1)}\right)\cdots\left(\phi\circ S_1,s_{k_1}^{(1)}\right)\cdots\left(\phi\circ S_1,s_1^{(n)}\right)\cdots\left(\phi\circ S_1,s_{k_n}^{(n)}\right)$$
$$= ((\phi\circ p)[\phi\circ S_1,\ldots,\phi\circ S_n],s)$$

where

$$s = t\left[\left(s_1^{(1)},\ldots,s_{k_1}^{(1)}\right),\ldots,\left(s_1^{(n)},\ldots,s_{k_n}^{(n)}\right)\right].$$

Now assume that $S \in K\langle\langle T_\Gamma\rangle\rangle$ is polynomially presentable, i.e. there exists a finite list $S_1,\ldots,S_n \in K\langle\langle T_\Gamma\rangle\rangle$ so that

$$S = p[S_{\mu_1},\ldots,S_{\mu_k}] \text{ and } \tau^{-1}S_i = r_i\left[S_{j_1},\ldots,S_{j_{\lambda_i}}\right]$$

for some polynomials $p \in K\langle T_\Gamma(X_k)\rangle$, $r_i \in K\langle T_\Gamma(X_{\lambda_i})\rangle$ and $\mu_1,\ldots,\mu_k$, $j_1,\ldots,j_{\lambda_i} \in \{1,2,\ldots,n\}, 1 \le i \le n$. Then $\phi\circ S = (\phi\circ p)[\phi\circ S_{\mu_1},\ldots,\phi\circ S_{\mu_k}]$ and $\tau^{-1}(\phi\circ S_i) = (\phi\circ r_i)\left[\phi\circ S_{j_1},\ldots,\phi\circ S_{j_{\lambda_i}}\right]$ and so $\phi\circ S$ is again a polynomially presentable series. □

By Proposition 4 and the remark made after the definition of polynomially presentable tree series, we have $REC(K,\Gamma) \subseteq LP(K,\Gamma) \subseteq PP(K,\Gamma)$. Next we show that $PP(K,\Gamma) - REC(K,\Gamma) \neq \emptyset$.

Proposition 7. *The series height* : $T_\Gamma \to \mathbb{N}$ *is polynomially presentable.*

Proof. Let $\mathcal{A}$ be the subalgebra of $\mathbb{N}\langle\langle T_\Gamma\rangle\rangle$ generated by the set $\{\mathbf{1}, height\}$, where $\mathbf{1}$ is the tree series over K and Γ whose all coefficients are equal to 1. Certainly, $height \in \mathcal{A}$. Let us show that $\mathcal{A}$ is invariant. Since, for every $\tau \in P_\Gamma, \tau^{-1}(\mathbf{1}) = \mathbf{1}$, it is sufficient to show that, for every $\tau \in P_\Gamma$, there is a polynomial $p \in \mathbb{N}\langle T_\Gamma(X_2)\rangle$ such that $\tau^{-1}(height) = p[\mathbf{1}, height]$.

We distinct the cases:

Case 1 $height(\tau) \ge |\tau|$. Then

$$\tau^{-1}height = \sum_{k=1}^{n}(height(\tau) - |\tau| - height(t_k))\,t_k + |\tau|\cdot\mathbf{1} + height$$

where $t_1,\ldots,t_n$ are all the trees verifying

$$height(t_k) \le height(\tau) - |\tau|.$$

Case 2 $height(\tau) < |\tau|$. Then it holds

$$\tau^{-1}height = |\tau|\cdot\mathbf{1} + height.$$

Hence, in any case $\tau^{-1}height \in \mathcal{A}$, as claimed. □

Corollary 8. $PP(\mathbb{N},\Gamma) - REC(\mathbb{N},\Gamma) \neq \emptyset$.

Proof. We only have to combine the previous result together with the fact that height is a non-recognizable tree series. □

In [Bo1], linearly presentable series are obtained as matrix representations and as behaviours of the so called tree modules. On the other hand, when K is a field, recognizable and linearly presentable series coincide (cf. [BA]).

It is an open question whether $LP(K,\Gamma) - REC(K,\Gamma) \neq \emptyset$ or $PP(K,\Gamma) - LP(K,\Gamma) \neq \emptyset$ for semirings which are not fields.

5 Almost Presentable Tree series

We define the tree series $S, S' : T_\Gamma \to K$ to be *almost equal* and write $S \equiv S'$ whenever $(S,t) = (S',t)$ for all but a finite number of t's.

The above equivalence relation is compatible with sum, scalar product and derivation, i.e.

$$S_i \equiv S_i' \ (i = 1,2), \quad S \equiv S', \quad \lambda \in K, \tau \in P_\Gamma$$

imply

$$S_1 + S_2 \equiv S_1' + S_2', \quad \lambda S \equiv \lambda S', \quad \tau^{-1}S \equiv \tau^{-1}S'.$$

Call a series $S \in K\langle\langle T_\Gamma \rangle\rangle$ *almost linearly* presentable whenever there is a finite list of series $S_1, \ldots, S_n \in K\langle\langle T_\Gamma \rangle\rangle$ such that $S \equiv \lambda_1 S_1 + \cdots + \lambda_n S_n$ for some $\lambda_1, \ldots, \lambda_n \in K$ and for all $\tau \in P_\Gamma$ and $i = 1, \ldots, n$ we have $\tau^{-1}S_i \equiv \mu_1 S_1 + \cdots + \mu_n S_n$ for some $\mu_1, \ldots, \mu_n \in K$.

The tree series *height* is almost linearly presentable since for all $\tau \in P_\Gamma$ it holds

$$\tau^{-1}height \equiv |\tau| \cdot \mathbf{1} + height.$$

Hence the class $ALP(K,\Gamma)$ *of* almost linearly presentable series properly contains that of almost recognizable tree series.

Moreover $ALP(K,\Gamma)$ is an invariant subalgebra of $K\langle\langle T_\Gamma \rangle\rangle$.

References

[BA] S. Bozapalidis, A. Alexandrakis: Représentations Matricielles des Séries D' Arbre Reconnaisables, Informatique the"orique et Applications, vol 23, n° 4(1989), 449-459.

[Bo1] S. Bozapalidis, Representable Tree series, Fundamenta Informaticae 21 (1994), 367-389.

[Bo2] S. Bozapalidis, Equational Elements in Additive Algebras, Theoretical Computer Science 32(1999), 1-33.

[BR] J. Berstel, C. Reutenauer, Formal Power Series on Trees, Theoretical Computer Science 18(1982), 115-142.

[SS] A. Salomaa, A. Soittola, Automata Theoretic Aspects of Formal Power Series, Springer (1997).

[KS] W. Kuich, A. Salomaa, Semirings, Automata, Languages, Springer (1986).

[GS1] F. Gécseg, M. Steinby, Tree Automata, Akademiai Kiado, Budapest (1984).

[GS2] F. Gécseg, M. Steinby, Tree Languages, Handbook of Formal Languages, vol. 3(1997), 1-68.

Received December, 2004

Acta Cybernetica 17 (2006) 471–483.

On the Complete Axiomatization for Prefix Iteration modulo Observation Congruence*

Taolue Chen† Tingting Han‡ and Jian Lu‡

Abstract

Prefix iteration is a variation on the original binary version of the Kleene star operation P^*Q, obtained by restricting the first argument to be an atomic action. Aceto and Ingólfsdóttir provided an axiom system for observation congruence over basic CCS with prefix iteration. However hitherto the only direct completeness proof given for such a system is very long and technical. In this paper, we provide a new proof for the completeness of the axiom system in [3], which is a considerable simplification comparing to the original proof. Thus the open problem to find a direct completeness proof is closed.
Key Words: Process Algebra, Prefix Iteration, Observation Congruence, Axiomatization, Completeness.

1 Introduction

Kleene [13] defined a binary operator -*- in the context of finite automata, called *Kleene star* or *iteration*. Intuitively, the expression p^*q yields a solution for the recursive equation $X = p.X + q$. In other words, p^*q can choose to execute either p, after which it evolves into p^*q again, or q, after which it terminates. An advantage of the Kleene star is that on the one hand it can express recursion, but that on the other hand one can capture this operator in equational laws. Hence, one does not need meta-principles, such as Milner's *Unique Fixpoint Induction Principle* [15]. Kleene formulated several equations for this operator, e.g. $x^*y = x(x^*y) + y$.

The research literature on process theory has witnessed a resurgence of the interest in the study of Kleene star-like operations (cf. e.g. the papers [1, 2, 3, 6, 8]).

*This work is partially supported by NNSFC (60233010, 60273034, 60403014) and 973 Program of China (2002CB312002).

†Corresponding author. CWI, Department of Software Engineering, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands; and State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, Jiangsu, P.R.China, 210093 *E-mail address*: Taolue.Chen@cwi.nl. The author is partially supported by the Dutch BSIK/BRICKS Project (Basic Research in Informatics for Creating the Knowledge Society).

‡State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, Jiangsu, P.R.China, 210093.

Some researchers have studied the possibility of giving finite equational axiomatization of the bisimulation-like equivalence [14, 17] over simple process algebras that include variations on Kleene's star operation.

First of all, Sewell [18] shows that there does not exist a complete finite equational axiomatization for BPA_δ [5] with the Kleene star modulo strong bisimulation. In the light of this result, one way to obtain an equational axiomatization is to restrict the range of the terms that might occur at the left-hand side of the binary Kleene. The paper [6] might be the starting point of the work following this line. In that reference, Fokkink proposes a *finite*, complete equational axiomatization of strong bisimulation equivalence for $\mathrm{BCCS}^{p*}(A)$, that is, the language obtained by extending the fragment of Milner's CCS [14] containing the basic operations needed to express finite synchronization trees with *prefix iteration* a^*x, where a ranges over the atomic actions. Aceto and Groote [1] generalize this result to *string* iteration w^*x, where w ranges over strings of atomic actions whose length is smaller than some positive natural number N. Aceto and Ingólfsdóttir study prefix iteration in the presence of the silent step τ, in Milner's observation congruence. They extend the axiomatization from [6] with two standard equations for the silent step, and with three new equations which describe the interplay between the silent step and prefix iteration. Moreover, the completeness of their equational axiomatization w.r.t. observation congruence is shown. By term rewriting techniques, in [7] Fokkink presents a considerably shorter completeness proof for prefix iteration together with the silent step in rooted branching bisimulation equivalence from van Glabbeek and Weijland [12] in the setting of BPA [5] with the deadlock δ and empty process ϵ. In an unpublished paper, van Glabbeek shows that the completeness result in observation congruence from [3] follows from the completeness result in rooted branching bisimulation. The combination of the results of Fokkink and van Glabbeek leads to a considerably shorter completeness proof for prefix iteration in observation congruence than the one presented in [3]. As a conclusion, Aceto, Fokkink, van Glabbeek and Ingólfsdóttir have merged their three papers into one paper [2], which deals at once with weak, branching, delay, and η-bisimulation. Among other things, this paper presents a self-contained completeness proof for prefix iteration modulo rooted branching bisimulation.

However, to our knowledge, all the efforts to give a *direct* proof of the completeness theorem for prefix iteration in Milner's *observation congruence* [14] which is simpler than the one presented in [3] have failed. Let us quote what the researchers who are active in this area said:

- In [7], Fokkink wrote: "... This paper results from an attempt to try and shorten the long and technical completeness proof in [3]. Although this attempt was unsuccessful for observation congruence, it did yield a a considerably shorter completeness proof for prefix iteration together with the silent step in rooted branching bisimulation equivalence from van Glabbeek and Weijland [12]...".

- In [2], Aceto, Fokkink, van Glabbeek and Ingólfsdóttir wrote: "... All the authors' attempts to obtain a direct proof of the completeness theorem for

weak congruence which is simpler than the one presented in [3] have been to no avail...". Note that here "weak congruence" refers to observation congruence in this paper.

This paper aims at giving a contribution to the study of complete equational axiomatization for Kleene star-like operations (concretely speaking, prefix iteration) from the point of view of process theory. We are motivated to shorten the long and technical completeness proof in [3], that is, to close the open problem in the sense of obtaining a direct proof of the completeness theorem for observation congruence which is simpler than Aceto and Ingólfsdóttir's. Following [3], we work on the language $\mathrm{BCCS}^{p*}(A_\tau)$. The axiom system for observation congruence has appeared in that reference, therefore our contribution lies in a much simpler and shorter proof. The main techniques used in this paper are standard. The source of simplicity, in our opinion, results from the extensive application of the so-called *Absorption Lemma* and the avoidance of well-known Hennessy Lemma [14]. Below we will examine this in more detail. It is worth pointing out that the following observation is not completely new, since it has been made by Fu and Yang in [10] for a language of mobile processes, π-calculus [16]. In particular, the promotion lemma is inspired by [10].

In the standard proof of the completeness theorem for observation congruence on finite CCS processes [14], one verifies first that every normal form process is provably equivalent to a saturated normal form process using the three τ-laws. Recall that a process P is saturated if, for each action α, one has that $P \xrightarrow{\alpha} P'$ whenever $P \stackrel{\alpha}{\Rightarrow} P'$. It follows that P is in saturated normal form if and only if whenever $P \stackrel{\alpha}{\Rightarrow} P'$ then $\alpha.P'$ is a summand of P. Now if P and Q are weakly congruent saturated normal form processes and $P \xrightarrow{\alpha} P'$ then $Q \stackrel{\alpha}{\Rightarrow} Q'$ for some Q' such that $Q' \approx P'$, where $\approx$ denotes weak bisimulation equivalence. By saturation, $Q \xrightarrow{\alpha} Q'$ and therefore $\alpha.Q'$ is a summand of Q. If, and this is a nontrivial if, we can deduce by the induction hypothesis that $\alpha.P'$ is provably equal to $\alpha.Q'$, which is much weaker than saying that P' is provably equal to Q', then we can conclude that every summand of P is provably equal to a summand of Q, and vice versa. This gives us the required completeness.

If one is only interested in a completeness proof, then the notion of saturated process is not needed. What is really necessary is the following *saturation property*, which is expressed by the *Absorption Lemma* in this paper.

If $P \stackrel{\alpha}{\Rightarrow} P'$ and P is in normal form, then P and $P + \alpha.P'$ are provably equal.

From the point of view of obtaining complete axiomatizations, the role of the saturation property is to relate operational semantics to equational rewriting. A careful examination of the role of the Hennessy Lemma in the completeness proof for CCS shows that what it really comes down to is the following property:

If $P \approx Q$ then either $\tau.P = Q$, or $P = Q$, or $P = \tau.Q$ is provable.

So the Hennessy Lemma helps to transfer a semantic statement to a proof theoretical one. As a matter of fact, as far as completeness is concerned, the following

weaker property is all one needs:

$$\text{If } P \approx Q \text{ then } \tau.P = \tau.Q \text{ is provable.}$$

In this paper, following [10], we call this the *promotion property*, expressed by the Promotion Lemma (Lemma 9), which relates behavioral semantics to equational rewriting. It promotes a pair of semantically equivalent processes to a pair of proof theoretically equal processes. Just by the *Absorption Lemma* and the *Promotion Lemma*, we circumvent the Hennessy Lemma to obtain the completeness. It is fair to say that actually in [2], Aceto et al. provided a similar property as promotion lemma (see [2], Proposition 4.3). However, there they focused on rooted branching bisimulation, and the motivation and details are quite different from ours.

Note that, as in [3] and unlike [2], we only focus on completeness rather than ω-completeness (i.e. completeness for equality of *open* terms over the signature of $\mathrm{BCCS}^{p*}(A_\tau)$). This is not a very serious shortcoming because of two reasons: (1) As in [3], using a technique due to Groote [11], it is not difficult to show ω-completeness. (2) We can provide the proof of ω-completeness by a minor modification on our proof, just as in [2]. We avoid doing this just because we intend to help the readers evade some unnecessary details and pay their full attention to the essence of our arguments.

The rest of the paper is organized as follows: Some preliminaries are reviewed in the following section. In Section 3, the simplified proof for completeness is presented. The paper is concluded with Section 4, where also related work is discussed.

2 Preliminaries

We assume a non-empty, countable set A of observable actions not containing the distinguished symbol τ. Following Milner, the symbol τ will be used to denote an internal, unobservable action of a system. We define $A_\tau \stackrel{\text{def}}{=} A \cup \{\tau\}$, and use a, b to range over A and α, β to range over A_τ. Note that we follow this convention strictly, so $a \neq \tau$ for any $a \in A$. We also assume a countably infinite set of process variables V, ranged over by x, y, z, that is disjoint from A_τ.

The language of basic CCS with prefix iteration, denoted by $\mathrm{BCCS}^{p*}(A_\tau)$, is given by the following BNF grammar:

$$P ::= x \mid 0 \mid \alpha.P \mid P + P \mid \alpha^* P$$

where $x \in V$ and $\alpha \in A_\tau$. The set of closed terms, i.e. terms that do not contain occurrences of process variables, generated by the above grammar, will be denoted by $\mathbf{T}(\mathrm{BCCS}^{p*}(A_\tau))$, while the set of open terms will be denoted by $\mathbb{T}(\mathrm{BCCS}^{p*}(A_\tau))$. We shall use P, Q (possibly subscripted and/or superscripted) to range over $\mathbb{T}(\mathrm{BCCS}^{p*}(A_\tau))$.

The operational semantics for the language $\mathrm{BCCS}^{p*}(A_\tau)$ is given by the labelled transition system $\langle \mathbf{T}(\mathrm{BCCS}^{p*}(A_\tau)), \{\stackrel{\alpha}{\rightarrow} \mid \alpha \in A_\tau\}\rangle$, where each transition relation

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \quad \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \quad \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

$$\frac{}{\alpha^*P \xrightarrow{\alpha} \alpha^*P} \quad \frac{P \xrightarrow{\alpha} P'}{\beta^*P \xrightarrow{\alpha} P'}$$

Figure 1: Operational Semantics.

$\xrightarrow{\alpha}$ is the least binary relation that satisfies the rules in **Fig.** 1. Following Milner [14], the derived transition relation $\Rightarrow$ is defined as the reflexive, transitive closure of $\xrightarrow{\tau}$, and $\stackrel{\alpha}{\Rightarrow}$, $\stackrel{\hat{\alpha}}{\Rightarrow}$ are defined in the standard way as follows:

$$\stackrel{\alpha}{\Rightarrow} \stackrel{def}{=} \Rightarrow\xrightarrow{\alpha}\Rightarrow$$

$$\stackrel{\hat{\alpha}}{\Rightarrow} \stackrel{def}{=} \begin{cases} \Rightarrow & \text{if } \alpha = \tau \\ \stackrel{\alpha}{\Rightarrow} & \text{otherwise.} \end{cases}$$

Now, we introduce some behavioral equivalences studied in this paper.

Definition 1. *(Weak Bisimulation) A binary relation $\mathcal{R}$ over $\mathbf{T}(BCCS^{p*}(A_\tau))$ is a* weak bisimulation *if it is symmetric and whenever $P\mathcal{R}Q$ and $P \xrightarrow{\alpha} P'$, then there exists some Q' s.t. $Q \stackrel{\hat{\alpha}}{\Rightarrow} Q'$ and $P'\mathcal{R}Q'$.*

Two process terms P, Q are observation equivalent, denoted by $P \approx Q$, if there exists a bisimulation $\mathcal{R}$ s.t. $P\mathcal{R}Q$.

As is well-known [14], $\approx$ is an equivalence relation. However, it is not a congruence w.r.t. the alternative composition operation and the prefix iteration operation.

Definition 2. *(Observation Congruence) For all process terms $P, Q \in \mathbf{T}(\mathrm{BCCS}^{p*}(A_\tau))$, $P \simeq Q$ iff for any $\alpha \in A_\tau$,*

- *If $P \xrightarrow{\alpha} P'$, then there exists some Q' s.t. $Q \stackrel{\alpha}{\Rightarrow} Q'$ and $P' \approx Q'$.*
- *If $Q \xrightarrow{\alpha} Q'$, then there exists some P' s.t. $P \stackrel{\alpha}{\Rightarrow} P'$ and $P' \approx Q'$.*

Definition 3. *For all $P, Q \in \mathbb{T}(BCCS^{p*}(A_\tau))$, $P \simeq Q$ iff $P\sigma \simeq Q\sigma$ if for every substitution $\sigma : V \to \mathrm{BCCS}^{p*}(A_\tau)$.*

Lemma 4. *The relation $\simeq$ is the largest congruence contained in $\approx$.*

Proof. cf. [2], Proposition 2.8. □

Lemma 5. *Let $a, b \in A$. If $a^*P \approx b^*Q$, then $a = b$.*

Proof. cf. [2], Lemma 2.11. □

A1	$x+y$	$=$	$y+x$
A2	$(x+y)+z$	$=$	$x+(y+z)$
A3	$x+x$	$=$	x
A4	$x+0$	$=$	x
PA1	$a.(a^*x)+x$	$=$	a^*x
PB1	$a^*(a^*x)$	$=$	a^*x

Figure 2: The Axiom System $\mathcal{F}$.

T1	$\alpha.\tau.x$	$=$	$\alpha.x$
T2	$\tau.x$	$=$	$\tau.x+x$
PT1	τ^*x	$=$	$\tau.x$
PT2	$\tau.(a^*x)$	$=$	$a^*(\tau.(a^*x))$
AT3	$a.(x+\tau.y)$	$=$	$a.(x+\tau.y)+a.y$
PT3	$a^*(x+\tau.y)$	$=$	$a^*(x+\tau.y+a.y)$

Figure 3: τ Laws for Observation Congruence.

3 Completeness of the Axiomatization

The main aim of this paper is to give a proof for completeness of the equational axiom system in [2, 3] for observation congruence. We first present the axiom system. The axiom system $\mathcal{F}$ is the one that is shown in [6] to characterize strong bisimulation over $\mathbf{T}(\mathrm{BCCS}^{p*}(A))$, which is given in **Fig.** 2. In addition, [3] extends it with two of Milner's standard τ-laws and three auxiliary equations that describe the interplay between the silent nature of τ and prefix iteration, which is reported in **Fig.** 3. We let $\mathcal{E}$ denote the system $\mathcal{F}$ together with these laws.

For an axiom system $\mathcal{T}$, we write $\mathcal{T} \vdash P = Q$ iff the equation $P = Q$ is provable from the axiom system $\mathcal{T}$ using the rules of equational logic. Often for convenience, we omit $\mathcal{T}$ and abbreviate it as $\vdash P = Q$. We write $P \stackrel{X}{=} Q$ as a short-hand for $\mathrm{A1}, \mathrm{A2}, X \vdash P = Q$. We use $P =_{AC} Q$ to denote that P and Q are equal modulo associativity and commutativity of $+$, i.e. $\mathrm{A1}, \mathrm{A2} \vdash P = Q$.

For $I = \{i_1, \ldots, i_n\}$ a finite index set, we write $\sum_{i \in I} P_i$ for $P_{i_1} + \cdots + P_{i_n}$. By convention, if $I = \emptyset$, then $\sum_{i \in I} P_i$ stands for 0.

Soundness of the system is shown in [3]. The remainder of the section is devoted to an alternative proof of completeness w.r.t. [3]. As usual, we first identify a subset of process terms of a special form, which will be convenient in the proof of the completeness result for observation congruence. Following a long-established tradition of the literature on process theory, we shall refer to these terms as *normal forms*. The set of normal forms we are after is the smallest subset of process terms including process terms having one of the following two forms:

$$\sum_{i \in I} \alpha_i.P_i \quad \text{or} \quad a^*(\sum_{i \in I} \alpha_i.P_i)$$

where the term P_i are themselves normal forms, and I, J are finite index sets. (Recall that the empty sum represents 0).

Lemma 6. *Each term can be proven equal to a normal form using equations* A4, PA1, PB1.

Proof. cf. [2], Lemma 4.1. □

Note 7. PT1 is a powerful equation and is introduced in [4] under the name of "*Fair Iteration Rule*", it is an equational formulation of Koomen's *Fair Abstraction Rule* [5]. By using PT1, τ^*-like terms can be excluded from normal forms.

In the proof of the completeness result to come, we shall make use of a weight function $w : \mathbf{T}(\mathrm{BCCS}^{p*}(A_\tau)) \to \mathbb{N}$. This is defined by structured induction on terms as follows:

$$\begin{array}{lcllcl} w(0) & = & 0 & w(\alpha.P) & = & w(P)+1 \\ w(P+Q) & = & w(P)+w(Q)+1 & w(\alpha^*P) & = & w(P)+1 \end{array}$$

Lemma 8. *(Absorption Lemma) For any* $P, Q \in \mathbf{T}(\mathrm{BCCS}^{p*}(A_\tau))$, *if* $P \stackrel{\alpha}{\Rightarrow} P'$, *then* $\vdash P = P + \alpha.P'$.

Proof. Standard result. See [14]. □

Lemma 9. *(Promotion Lemma) For all* $P, Q \in \mathbf{T}(\mathrm{BCCS}^{p*}(A_\tau))$, *if* $P \approx Q$, *then* $\vdash \tau.P = \tau.Q$.

Proof. By Lemma 6, it is sufficient to prove the statement of the lemma for weakly bisimilar normal forms P and Q. So let us assume that P and Q are weakly bisimilar normal forms and we show that $\tau.P = \tau.Q$ by induction on the sum of the weights of P and Q. Recall that normal forms can take the following two forms:

$$\sum_{i\in I} \alpha_i.P_i \quad \text{or} \quad a^*(\sum_{i\in I} \alpha_i.P_i)$$

where the P_i are themselves normal forms. So, in particular, P and Q have one of these forms. By symmetry, it is sufficient to deal with the following three cases:

1. $P = \sum_{i\in I} \alpha_i.P_i$ and $Q = \sum_{j\in J} \beta_j.Q_j$.
2. $P = a^*(\sum_{i\in I} \alpha_i.P_i)$ and $Q = b^*(\sum_{j\in J} \beta_j.Q_j)$.
3. $P = \sum_{i\in I} \alpha_i.P_i$ and $Q = a^*(\sum_{j\in J} \beta_j.Q_j)$.

We treat these three cases separately.

1. CASE: $P = \sum_{i\in I} \alpha_i.P_i$ and $Q = \sum_{j\in J} \beta_j.Q_j$. Consider a summand $\alpha_i.P_i$ of P. It gives rise to a transition $P \stackrel{\alpha_i}{\to} P_i$, and hence since $P \approx Q$, we can distinguish two subcases in the proof:

- $\alpha_i \neq \tau$. Then we have $Q \stackrel{\alpha_i}{\Rightarrow} Q'$ and $P_i \approx Q'$. Since both P_i and Q' are in normal form and $w(P_i) + w(Q') < w(P) + w(Q)$, by the induction hypothesis, $\vdash \tau.P_i = \tau.Q'$. It follows that

$$\begin{aligned} \vdash Q &= Q + \alpha_i.Q' && \text{(Lemma 8)} \\ &\stackrel{\text{T1}}{=} Q + \alpha_i.\tau.Q' \\ &= Q + \alpha_i.\tau.P_i \\ &\stackrel{\text{T1}}{=} Q + \alpha_i.P_i \end{aligned}$$

  Thus we can obtain that $\vdash Q = Q + \sum_{\alpha_i \neq \tau} \alpha_i.P_i$.

- $\alpha_i = \tau$. Then either $Q \stackrel{\tau}{\Rightarrow} Q'$ or $Q = Q'$. In both cases, $P_i \approx Q'$. By the induction hypothesis, $\vdash \tau.P_i = \tau.Q'$. For the first case, it can be easily shown that $\vdash Q = Q + \tau.P_i$. In the second case one has $\vdash \tau.P_i = \tau.Q$.

In summary, for $i \in I$, we have either $\tau.Q = \alpha_i.P_i$ or $Q = Q + \alpha_i.P_i$. It follows that $\vdash Q + P = Q + \sum_{\{i|\alpha_i \neq \tau\}} \alpha_i.P_i + \sum_{\{i|\alpha_i = \tau\}} \alpha_i.P_i = Q + \sum_{k \in K} \tau.Q$ for some index set $K = \{k \mid \alpha_k = \tau \text{ and } P_i \approx Q \text{ and } k \in I\}$. Consequently we have $\vdash \tau.(P + Q) = \tau.(Q + \sum_{k \in K} \tau.Q) \stackrel{\text{T2,A3}}{=} \tau.Q$. Symmetrically $\vdash \tau.(P + Q) = \tau.P$. Therefore $\vdash \tau.P = \tau.Q$.

2. CASE: $P = a^*(\sum_{i \in I} \alpha_i.P_i)$ and $Q = b^*(\sum_{j \in J} \beta_j.Q_j)$. First of all, note that by Lemma 5, it must be the case that $a = b$. For convenience, we write $P_1 = \sum_{i \in I} \alpha_i.P_i$ and $Q_1 = \sum_{j \in J} \beta_j.Q_j$. Then $P = a^*P_1$ and $Q = a^*Q_1$. For each transition $P \stackrel{\alpha_i}{\rightarrow} P_i$, since $P \approx Q$, we can distinguish three cases in the proof:

   - $P_1 \stackrel{\alpha_i}{\rightarrow} P_i$ and $\alpha_i \neq a, \tau$. Then it must be that $Q_1 \stackrel{\alpha_i}{\Rightarrow} Q'$ and $P_i \approx Q'$. By the induction hypothesis, $\vdash \tau.P_i = \tau.Q'$. Following the same lines as in CASE (1), we have $\vdash Q_1 = Q_1 + \alpha_i.P_i$. Thus we can obtain that $\vdash Q_1 = Q_1 + \sum_{\{i|\alpha_i \neq \tau, a\}} \alpha_i.P_i$.

   - $P_1 \stackrel{a}{\rightarrow} P_i$. Then there are three subcases:

     (i) $Q_1 \stackrel{a}{\Rightarrow} Q'$ and $P_i \approx Q'$. Then by the induction hypothesis, $\vdash \tau.P_i = \tau.Q'$. It follows that $\vdash Q_1 = Q_1 + a.P_i$;

     (ii) $Q \stackrel{a}{\rightarrow} Q$, $Q_1 \stackrel{\tau}{\Rightarrow} Q'$ and $P_i \approx Q'$. Then by the induction hypothesis, $\vdash \tau.P_i = \tau.Q'$. Moreover, we have $\vdash Q_1 = Q_1 + \tau.P_i$. Thus we can show that $\vdash Q_1 = Q_1 + \sum_{k \in K_1} \tau.P_k$ for some index set $K_1 \subseteq \{k \mid \alpha_k = a \text{ and } k \in I\}$;

     (iii) $Q \stackrel{a}{\rightarrow} Q$ and $P_i \approx Q$. Then by the induction hypothesis, $\vdash \tau.P_i = \tau.Q$. Thus $\vdash a.P_i = a.\tau.P_i = a.\tau.Q = a.Q$.

   Therefore, from (i)-(iii), we can conclude that $\vdash Q_1 + \sum_{k \in K_1 \cup K_2} a.P_k = Q_1 + \sum_{\{i|\alpha_i = a\}} a.P_i$ for the index set K_1 and some index set K_2, where

$K_1, K_2 \subseteq \{k \mid \alpha_k = a \text{ and } k \in I\}$ and $K_1 \cap K_2 = \emptyset$. Moreover, $\vdash Q_1 = Q_1 + \sum_{k \in K_1} \tau.P_k$ and $\vdash \tau.P_k = \tau.Q$ for $k \in K_2$. In the sequel, we write $K = K_1 \cup K_2$.

- $P_1 \xrightarrow{\tau} P_i$. Then either $Q_1 \stackrel{\tau}{\Rightarrow} Q'$ or $Q = Q'$ (note that $a \neq \tau$). In both cases, $P_i \approx Q'$. By the induction hypothesis, $\vdash \tau.P_i = \tau.Q'$. For the first case, it can be easily shown that $\vdash Q_1 = Q_1 + \tau.P_i$. In the second case one has $\vdash \tau.P_i = \tau.Q$. Thus $\vdash Q_1 + \sum_{l \in L} \tau.Q = Q_1 + \sum_{\{i \mid \alpha_i = \tau\}} \alpha_i.P_i$ for some index set $L = \{l \mid \alpha_l = \tau \text{ and } P_l \approx .Q \text{ and } l \in I\}$.

Combining the above three cases, actually we have $\vdash Q_1 + P_1 = Q_1 + \sum_{l \in L} \tau.Q + \sum_{k \in K_1 \cup K_2} a.P_k$. Moreover

$$\vdash \quad Q_1 \quad = Q_1 + \sum_{k \in K_1} \tau.P_k \tag{1}$$

$$\vdash \quad \tau.P_k \quad = \tau.Q \quad \text{for } k \in K_2 \tag{2}$$

It follows that

$$\begin{aligned}
\vdash a^*(P_1 + Q_1) \quad &= \quad a^*(Q_1 + \sum_{l \in L} \tau.Q + \sum_{k \in K_1 \cup K_2} a.P_k) \\
&\stackrel{(1)}{=} a^*(Q_1 + \sum_{l \in L} \tau.Q + \sum_{k \in K_1} a.P_k + \sum_{k \in K_2} a.P_k + \sum_{k \in K_1} \tau.P_k) \\
&\stackrel{\text{PT3}}{=} a^*(Q_1 + \sum_{l \in L} \tau.Q + \sum_{k \in K_1} \tau.P_k + \sum_{k \in K_2} a.P_k) \\
&\stackrel{(1)}{=} a^*(Q_1 + \sum_{l \in L} \tau.Q + \sum_{k \in K_2} a.P_k)
\end{aligned}$$

Now, we have to distinguish two cases:

- If $K_2 \neq \emptyset$, then we have

$$\begin{aligned}
\vdash a^*(P_1 + Q_1) \quad &= \quad a^*(Q_1 + \sum_{l \in L} \tau.Q + \sum_{k \in K_2} a.P_k) \\
&\stackrel{(2)}{=} a^*(Q_1 + \sum_{l \in L} \tau.Q + a.Q) \\
&\stackrel{\text{PA1}}{=} a^*(Q + \sum_{l \in L} \tau.Q)
\end{aligned}$$

  We proceed by considering the following two subcases:

  - $L = \emptyset$. Then $\vdash a^*(P_1 + Q_1) = a^*Q \stackrel{\text{PB1}}{=} Q$.

– $L \neq \emptyset$. Then it follows that

$$\begin{aligned} \vdash a^*(P_1 + Q_1) &= a^*(Q + \sum_{l \in L} \tau.Q) \\ &\overset{\text{T2}}{=} a^*(\tau.Q) \\ &= a^*(\tau.a^*Q_1) \\ &\overset{\text{PT2}}{=} \tau.Q \end{aligned}$$

- If $K_2 \neq \emptyset$, then actually $\vdash a^*(P_1 + Q_1) = a^*(Q_1 + \sum_{l \in L} \tau.Q)$. Similarly, we also proceed by considering the following two subcases:
  - $L = \emptyset$. Then $\vdash a^*(P_1 + Q_1) = a^*Q_1 = Q$.
  - $L \neq \emptyset$. Then it follows that

$$\begin{aligned} \vdash a^*(P_1 + Q_1) &= a^*(Q_1 + \sum_{l \in L} \tau.Q) \\ &\overset{\text{A3,T2}}{=} a^*(Q_1 + \tau.Q + a^*Q_1) \\ &\overset{\text{PA1}}{=} a^*(Q_1 + \tau.Q + a.a^*Q_1 + Q_1) \\ &\overset{\text{A3,PA1}}{=} a^*(\tau.Q + Q) \\ &\overset{\text{T2}}{=} a^*(\tau.a^*Q_1) \\ &\overset{\text{PT2}}{=} \tau.Q \end{aligned}$$

From the above, we conclude that either $\vdash \tau.Q = a^*(P_1 + Q_1)$ or $\vdash Q = a^*(P_1 + Q_1)$. Symmetrically $\vdash \tau.P = a^*(P_1 + Q_1)$ or $P = a^*(P_1 + Q_1)$. For each of four combinations, clearly, we have $\vdash \tau.P = \tau.Q$, possibly using T1.

3. CASE: $P = \sum_{i \in I} \alpha_i.P_i$ and $Q = a^*(\sum_{j \in J} \beta_j.Q_j)$. For convenience, we write $Q_1 = \sum_{j \in j} \beta_j.Q_j$. We proceed by examining to the two directions of observation congruence.

   - For each transition from P, we consider three subcases in the proof:
     – $P \xrightarrow{\alpha_i} P_i$ and $\alpha_i \neq \tau, a$. Since $P \approx Q$, this transition from P must be matched by a transition $Q_1 \overset{\alpha_i}{\Rightarrow} Q'$ and $P_i \approx Q'$.
     – $P \xrightarrow{a} P_i$. Since $P \approx Q$, this transition from P must be matched by transitions $Q \xrightarrow{a} Q$, $Q_1 \overset{\tau}{\Rightarrow} Q'$ and $P_i \approx Q'$, or $Q \xrightarrow{a} Q$ and $P_i \approx Q$, or $Q_1 \overset{a}{\Rightarrow} Q'$ and $P_i \approx Q'$.
     – $P \xrightarrow{\tau} P_i$. Since $P \approx Q$, this transition from P must be matched by transitions $Q_1 \overset{\tau}{\Rightarrow} Q'$ or just $Q = Q'$. In both cases, $P_i \approx Q'$.

     Following the same lines of CASE (2), we can conclude that either $\vdash \tau.Q = a^*(P + Q_1)$ or $\vdash Q = a^*(P + Q_1)$.
   - For the transition from Q, we only need to consider the following two situations:

- $Q_1 \xrightarrow{\beta_j} Q_j$. Since $P \approx Q$, it must be matched by a transition $P \stackrel{\hat{\beta}_j}{\Rightarrow} P'$ and $P' \approx Q_j$. Following the same line of CASE (1), we obtain that $\vdash \tau.(P + Q_1) = \tau.P$.
- $Q \xrightarrow{a} Q$. Since $P \approx Q$, it must be matched by a transition $P \stackrel{a}{\Rightarrow} P'$ and $P' \approx Q$. By the induction hypothesis, we have $\vdash \tau.P' = \tau.Q$. Clearly, it follows that $\vdash P = P + a.Q$.

Then because $\vdash P = P + a.Q = P + a.a^*(P + Q_1)$ or $\vdash P = P + a.Q = P + a.\tau.Q = P + a.a^*(P + Q_1)$, we have $\vdash P + Q_1 = P + Q_1 + a.a^*(P + Q_1) \stackrel{\text{PA1}}{=} a^*(P + Q_1)$. It follows that $\vdash \tau.Q = \tau.(P + Q_1) = \tau.P$.

The proof is complete. □

Theorem 10. *(Completeness) If $P \simeq Q$, then $\vdash P = Q$.*

Proof. Consider two process terms P and Q that are observation congruent. We shall show that $P + Q$ is provably equal to Q. Of course, by symmetry, $P + Q$ is also provably equal to P. Hence we obtain completeness.

To this end, note that by Lemma 6, P and Q may be proven equal to some normal forms using A4, PA1 and PB1. Possibly using equation PA1 again, we may therefore derive that $\vdash P = \sum_{i \in I} \alpha_i.P_i$ and $\vdash Q = \sum_{j \in J} \beta_j.Q_j$ for some finite index sets I, J. Consider a summand $\alpha_i.P_i$ of P. It gives rise to a transition $P \xrightarrow{\alpha_i} P_i$ and hence, since $P \simeq Q$, there exists some Q' such that $Q \stackrel{\alpha_i}{\Rightarrow} Q'$ and $P_i \approx Q'$. By Lemma 9, we have $\vdash \tau.P_i = \tau.Q'$. It follows that

$$\begin{aligned} \vdash Q &= Q + \alpha_i.Q' && \text{(Lemma 8)} \\ &\stackrel{\text{T1}}{=} Q + \alpha_i.\tau.Q' \\ &= Q + \alpha_i.\tau.P_i \\ &\stackrel{\text{T1}}{=} Q + \alpha_i.P_i \end{aligned}$$

Consequently, we have $\vdash Q = Q + \sum_{i \in I} \alpha_i.P_i = Q + P$. By symmetry, $\vdash P = P + Q$. Thus $\vdash P = P + Q = Q$. The proof is complete. □

4 Conclusion

Related Work. Besides the works we have pointed out in Section 1, we would like to mention some other related works, which are mainly dealing with the axiomatization of strong bisimilarity over BPA with Kleene star-like operation. In [15], Milner first studies iteration in strong bisimulation equivalence in a process algebra equivalent to $\text{BPA}_{\delta\epsilon}$ extended with the Kleene star. Bergstra, Bethke and Ponse [4] consider BPA with the Kleene star, and they suggest a finite equational axiomatization for this algebra. In [8], Fokkink and Zantema prove that this axiomatization is complete w.r.t. strong bisimulation equivalence. In [7], Fokkink presents a simpler and shorter completeness proof. In [9], Fokkink and Zantema

also study a rewrite system that stems from axioms for prefix iteration in $\mathrm{BPA}_{\delta\epsilon}$ and obtain a complete axiomatization.

Acknowledgement. We are grateful to Wan Fokkink for his careful reading of a draft of this paper and valuable comments. We also would like to thank the anonymous referees for their constructive criticism on a previous submission.

References

[1] L. Aceto and J. F. Groote. A complete equational axiomatization for MPA with string iteration. *Theor. Comput. Sci.*, 211(1-2):339–374, 1999.

[2] L. Aceto, R. J. van Glabbeek, W. Fokkink, and A. Ingólfsdóttir. Axiomatizing prefix iteration with silent steps. *Inf. Comput.*, 127(1):26–40, 1996.

[3] L. Aceto and A. Ingólfsdóttir. An equational axiomatization of observation congruence for prefix iteration. In M. Wirsing and M. Nivat, editors, *AMAST*, volume 1101 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 1996.

[4] J. A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *Comput. J.*, 37(4):243–258, 1994.

[5] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[6] W. Fokkink. A complete equational axiomatization for prefix iteration. *Inf. Process. Lett.*, 52(6):333–337, 1994.

[7] W. Fokkink. A complete axiomatization for prefix iteration in branching bisimulation. *Fundam. Inform.*, 26(2):103–113, 1996.

[8] W. Fokkink and H. Zantema. Basic process algebra with iteration: Completeness of its equational axioms. *Comput. J.*, 37(4):259–268, 1994.

[9] W. Fokkink and H. Zantema. Prefix iteration in basic process algebra: applying termination techniques. In A. Ponse, C. Verhoef, and S. van Vlijmen, editors, *Proceeding 2nd workshop on the ACP (ACP'95)*, Eindhoven, 1995.

[10] Y. Fu and Z. Yang. Tau laws for pi calculus. *Theor. Comput. Sci.*, 308(1-3):55–130, 2003.

[11] J. F. Groote. A new strategy for proving omega-completeness applied to process algebra. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 314–331. Springer, 1990.

[12] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.

[13] S. Kleene. Representation of events in nerve nets and finite automata. In *Automata studies*, pages 3–41. Princeton University Press, 1956.

[14] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[15] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviors. *Inf. Comput.*, 81(2):227–247, 1989.

[16] R. Milner, J. Parrow, and D. Walker. A calculus of mobile process, part I/II. 100:1–77, 1992.

[17] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[18] P. Sewell. Nonaxiomatisability of equivalences over finite state processes. *Ann. Pure Appl. Logic*, 90(1-3):163–191, 1997.

Received October, 2005

Acta Cybernetica **17** (2006) 485–511.

Relational Databases and Homogeneity in Logics with Counting

José María Turull Torres*

Abstract

We define a new hierarchy in the class of computable queries to relational databases, in terms of the preservation of equality of theories in fragments of first order logic with bounded number of variables with the addition of counting quantifiers (C^k). We prove that the hierarchy is strict, and it turns out that it is orthogonal to the TIME-SPACE hierarchy defined with respect to the Turing machine complexity. We introduce a model of computation of queries to characterize the different layers of our hierarchy which is based on the reflective relational machine of S. Abiteboul, C. Papadimitriou, and V. Vianu. In our model the databases are represented by their C^k theories. Then we define and study several properties of databases related to homogeneity in C^k getting various results on the change in the computation power of the introduced machine, when working on classes of databases with such properties. We study the relation between our hierarchy and a similar one which we defined in a previous work, in terms of the preservation of equality of theories in fragments of first order logic with bounded number of variables, but *without* counting quantifiers (FO^k). Finally, we give a characterization of the layers of the two hierarchies in terms of the infinitary logics $\mathcal{C}^k_{\infty\omega}$ and $\mathcal{L}^k_{\infty\omega}$, respectively.

Keywords: query languages, database machines, query computability, completeness of models, counting

1 Introduction

Given a relational database schema, it is natural to think about the whole class of queries which might be computed over databases of that schema. That is, if we do not restrict ourselves to a given implementation of certain query language on some computer, in the same way as the notion of computable function over the natural numbers was raised in computability theory. In [CH80], A. Chandra and D. Harel devised a formalization for that notion. They defined a *computable query*

*Massey University, Department of Information Systems, Information Science Research Centre, PO Box 756, Wellington, New Zealand E-mail: j.m.turull@massey.ac.nz

as a function over the class of databases of some given schema which is not only recursive but preserves isomorphisms as well. Isomorphism preservation is what formalizes the intuitive property of representation independence.

In [Tur01a, Tur04] a strict hierarchy was defined, in the class of computable queries ($\mathcal{CQ}$) of A. Chandra and D. Harel, in terms of the preservation of equivalence in bounded variable fragments of first order logic (FO), which we denote by FO^k. The logic FO^k is the fragment of FO which consists of the formulas with up to k different variables. We denote the whole hierarchy as $\mathcal{QCQ}^\omega$. For every natural k, the layer denoted as $\mathcal{QCQ}^k$ was proved to be a semantic characterization of the computation power of the reflective relational machine of [APV98] with variable complexity k (RRM^k). The RRM^k is a model of computation of queries to relational databases which has been proved to be incomplete, i.e., it does not compute all computable queries. Then, we defined and studied in [Tur01a, Tur04] several properties of relational databases, related to the notion of homogeneity in model theory [CK92], as properties which increase the computation power of the RRM^k when working on databases having those properties.

That research was enrolled in a very fruitful research program in the field of finite model theory considered as a theoretical framework to study relational databases. In that program different properties of the databases have been studied, which allow incomplete computation models to change their expressive power when computing queries on databases with those properties. Order [AV95, EF99], different variants of rigidity [DLW95, Tur96, Tur98], and different notions related to homogeneity [Tur01a, Tur04] are properties which turned out to be quite relevant to the expressive power of certain models of computation.

In the present paper, and following the same research program, we define a new hierarchy in the class of computable queries, which we denote as $\mathcal{QCQ}^{C^\omega}$. We define this hierarchy in terms of the preservation of equivalence in bounded variable logics with *counting quantifiers* (C^k). For every natural k, we denote as $\mathcal{QCQ}^{C^k}$ the layer of the hierarchy $\mathcal{QCQ}^{C^\omega}$ which consists of those queries that preserve equivalence in C^k. The logic C^k is obtained by adding quantifiers "there exists at least m different elements in the database such that..." to the logic FO^k for every natural m. The logic C^k has been deeply studied during the last decade [CFI92, Gro96, Hel96, Ott97].

Defining the classes $\mathcal{QCQ}^{C^k}$ appears to be rather natural, since in the definition of *computable query* of [CH80] the property of preservation of isomorphisms is essential, and, as it is well known, in finite databases isomorphism coincides with equivalence in first order logic. Moreover, it is also well known that for every natural k, the logic C^k is strictly weaker than FO. So, when we define subclasses of computable queries in terms of the preservation of C^k equivalence, for different values of k, in a certain way we are classifying queries according to the *different levels in the amount of information which we really need about the input database* to evaluate a given query on that database.

The hierarchy $\mathcal{QCQ}^{C^\omega}$ turns out to have quite similar structure and behavior as the hierarchy $\mathcal{QCQ}^\omega$ [Tur01a, Tur04]. The results of Sections 3 and 4 are analogous

to the results in [Tur01a, Tur04] regarding $\mathcal{QCQ}^\omega$, and their proofs follow a similar strategy. However, there is a very important difference in the expressiveness of the layers of both hierarchies, which is not surprising given the well known difference in expressive power between the logics FO^k and C^k. For every $k \geq 2$ the subclass $\mathcal{QCQ}^{C^k}$ is "big", whereas each subclass $\mathcal{QCQ}^k$ is "small", in a sense which we will make precise in Section 5 using results on asymptotic probabilities. Roughly, we can say that for every computable query q there is a query q' in the layer $\mathcal{QCQ}^{C^2}$ which is equivalent to q over *almost all* databases. And this is not true for any layer in $\mathcal{QCQ}^\omega$, and not even for the whole hierarchy.

We prove that the hierarchy $\mathcal{QCQ}^{C^\omega}$ is strict, and that it is strictly included in the class $\mathcal{CQ}$ of computable queries. Furthermore, it turns out to be *orthogonal* to the TIME-SPACE hierarchy defined with respect to Turing machine complexity, as it was also the case with the hierarchy $\mathcal{QCQ}^\omega$. Hence, we can define finer classifications in the class $\mathcal{CQ}$ by intersecting $\mathcal{QCQ}^{C^\omega}$ with the Turing machine complexity hierarchy (see [Tur01b] for a preliminary discussion of this approach). As an illustrating example, we include a classification of some problems in finite group theory at the end of Section 5. This may be derived from results in [KL99] and [KL00] together with our characterization of the layers of $\mathcal{QCQ}^\omega$ in terms of fragments of the infinitary logic $\mathcal{L}^\omega_{\infty\omega}$.

Having defined the different classes $\mathcal{QCQ}^{C^k}$ in a semantic way, we look next for a syntactic characterization of these classes in terms of a computation model. For that sake we define a machine which we call *reflective counting machine* with bounded variable complexity k (RCM^k), as a variant of the reflective relational machine of [APV98]. In our model, dynamic queries are formulas of C^k, instead of FO^k. In [Ott96] a similar model has been defined to characterize the expressibility of fixed point logics with counting terms, but it was based on the relational machine of [AV95], instead. Then we prove that for every natural k, the class $\mathcal{QCQ}^{C^k}$ characterizes exactly the expressive power of the machine RCM^k.

The model RCM^k turns out to be incomplete, i.e., there are computable queries which cannot be computed by any such machine. Then, we define several properties related to homogeneity (which are quite analogous to the properties studied in ([Tur01a], [Tur04]) regarding the model RRM^k), and we study the way in which the computation power of the model RCM^k changes when working on such classes of databases. Such properties are *C^k-homogeneity*, *strong C^k-homogeneity* and *pairwise C^k-homogeneity*. A database is *C^k-homogeneous* if the properties of every k-tuple in the database, up to automorphism, can be expressed by C^k formulas (i.e., whenever two tuples satisfy the same properties expressed by FO formulas with k variables and with counting quantifiers, then there is an automorphism of the database mapping each tuple onto each other). We prove that for every $k \geq 1$ there are queries whose restriction to C^k-homogeneous databases can be computed by RCM^k machines, whilst the same queries *cannot* be computed by any RCM^k on the *whole class* of databases of the given schema. A database is *strongly C^k-homogeneous* if it is C^r-homogeneous for every $r \geq k$. Here we show that, roughly speaking, for every $r > k \geq 1$, the class of queries whose restric-

tion to such classes of databases can be computed by RCM^r machines, strictly includes the class of queries whose restriction to classes of databases which are C^k-homogeneous but which are not strongly C^k-homogeneous can be computed by RCM^r machines. Concerning the third notion, we say that a class of databases is *pairwise C^k-homogeneous* if for every pair of databases in the class, and for every pair of k-tuples taken respectively from the domains of the two databases, if both k-tuples satisfy the same properties expressible by C^k formulas, then the two databases are isomorphic and there is an isomorphism mapping one tuple onto the other. We show that for every k, machines in RCM^k working on such classes achieve completeness provided the classes are recursive.

Considering that equivalence in C^k is decidable in polynomial time [Ott97], a very important line of research, which is quite relevant to complexity theory, is the identification of classes of graphs where C^k equivalence coincides with isomorphism. These classes are the classes which we define as *pairwise C^k-homogeneous*, and include the class of trees [IL90] and the class of planar graphs [Gro98b]. In the study of C^k-homogeneity we intend to generalize this approach by defining a formal framework, and by considering not only those "optimal" classes, but also other properties which, still not being so powerful as to equate C^k equivalence with isomorphism, do increase the computation power of the model RCM^k to an important extent.

In Section 5, we investigate the relationship between our classes $\mathcal{QCQ}^{C^k}$ and recursive fragments of the infinitary logic $C^\omega_{\infty\omega}$. We prove that for every natural k, the restriction of $\mathcal{QCQ}^{C^k}$ to Boolean queries characterizes the expressive power of $C^k_{\infty\omega}$ restricted to sentences with recursive classes of models. As a corollary, we get a characterization of the expressive power of the model RRM^k restricted to Boolean queries, for every natural k, in terms of the infinitary logic $\mathcal{L}^k_{\infty\omega}$. The characterization for the whole class of *relational machines* (RM) (and, hence, also of $RRM^{O(1)}$, given the equivalence of the two classes which was proved in [AV95]) in terms of the infinitary logic $\mathcal{L}^\omega_{\infty\omega}$ was proved in [AVV95], but the expressive power of each subclass of machines RRM^k in terms of the corresponding fragment of the infinitary logic was unknown up to the author's knowledge.

Some of the results presented here have been included in [Tur01b].

An extended abstract of this article has been published as [Tur02].

2 Preliminaries

Unless otherwise stated, in the present article we will follow the usual notation in finite model theory, as in [EF99]. We define a *relational database schema*, or simply *schema*, as a set of relation symbols with associated arities. We do not allow constraints in the schema, and we do not allow constant symbols either. If $\sigma = \langle R_1, \ldots, R_s \rangle$ is a schema with arities $r_1, \ldots, r_s$, respectively, a *database instance* or simply *database* over the schema σ, is a structure $I = \langle D^I, R^I_1, \ldots, R^I_s \rangle$ where D^I is a finite set which contains exactly all elements of the database, and for $1 \leq i \leq s$, R^I_i is a relation of arity r_i, i.e., $R^I_i \subseteq (D^I)^{r_i}$. We will often use $dom(I)$ instead of

D^I. We define the *size* of the database I as the cardinality of D^I, i.e., $|D^I|$. We will use $\simeq$ to denote the isomorphism relation. A *k-tuple* over a database I, for $k \geq 1$, is a tuple of length k formed from elements of $dom(I)$. We will denote a k-tuple of I by $\bar{a}_k$, or simply by $\bar{a}$. We use $\mathcal{B}_\sigma$ to denote the class of all *finite* databases of schema σ.

2.1 Computable Queries and Relational Machines

In this paper, we will consider *total* queries only. Let σ be a schema, let $r \geq 1$, and let R be a relation symbol of arity r. A *computable query of arity r and schema* σ [CH80], is a total recursive function $q^r : \mathcal{B}_\sigma \rightarrow \mathcal{B}_{\langle R \rangle}$ which preserves isomorphisms and such that for every database I of schema σ, $dom(q(I)) \subseteq dom(I)$. By preservation of isomorphisms we mean that for every $I, J \in \mathcal{B}_\sigma$ and for every isomorphism $f : dom(I) \rightarrow dom(J)$, $q(J) = f(q(I))$. A Boolean query is a 0-ary query. We denote the class of computable queries of schema σ as $\mathcal{CQ}_\sigma$, and $\mathcal{CQ} = \bigcup_\sigma \mathcal{CQ}_\sigma$. Relational machines ($RM$) have been introduced in [AV95]. A RM is a one-tape Turing machine (TM) with the addition of a *relational store* (rs) formed by a possibly infinite set of relations whose arity is *bounded* by some integer. The *only* way to access the relations in the rs is through FO (first order logic) formulas in the finite control of the machine. The input database as well as the output relation are in rs. In a transition of the machine one of these FO formulas can be evaluated in rs. The resulting relation is then assigned to some relation symbol of the appropiate arity in the rs. The *arity* of a given relational machine is the maximum number of variables (free or bound) of any formula in its finite control.

Reflective relational machines (RRM) have been introduced in [APV98] as an extension of RMs. In an RRM, FO queries are generated *during the computation* of the machine, and they are called *dynamic queries*. Each of these queries is written on a *query tape* and it is evaluated by the machine in one step. A further important difference to RM is that in RRM relations in the rs can be of *arbitrary arity*. New relations can be created in rs during a computation, and they can be used in building the dynamic queries. An integer index can be used in the formulas to name a relation, and if that relation does not exist in rs it is created with the appropriate arity. The *variable complexity* of an RRM is the maximum number of variables which may be used in the dynamic queries generated by the machine throughout any computation. We will denote as RRM^k, with $k \geq 1$, the sub-class of RRM with variable complexity k. Furthermore, we define $RRM^{O(1)} = \bigcup_{k \geq 1} RRM^k$.

In [AVV97] it was shown that $RRM^{O(1)} = RM$, i.e., that the class of queries which can be computed by reflective relational machines of bounded variable complexity is exactly the same as the class of queries which can be computed by relational machines. However, it is *not* known whether for every RRM of variable complexity $O(1)$ there exists an equivalent RM whose arity is the *same* as the variable complexity of the given RRM. Moreover, this is strongly believed to be *not* true (see [AVV95], particularly Remark 3.3, and [AV95]).

2.2 Finite Model Theory and Databases

We refer the reader to [EF99] and [Imm99] for an in-depth study of finite model theory, and to [AHV94] for the relation between databases and finite model theory. We will use the notion of a *logic* in a general sense. A formal definition would only complicate the presentation and is unnecessary for our work. The interested reader can see [Ebb85] for a formal study of a general framework of abstract logics. As usual in finite model theory, we will regard a logic as a language, that is, as a set of formulas (see [EF99, AHV94]). We will only consider signatures, or vocabularies, which are purely *relational*. We will always assume that the signature includes a symbol for equality. We consider *finite* structures only. Consequently, if $\mathcal{L}$ is a logic, the notion of *satisfaction*, denoted as $\models_{\mathcal{L}}$, will be related to only finite structures. If $\mathcal{L}$ is a logic and σ is a signature, we will denote by $\mathcal{L}_\sigma$ the class of formulas from $\mathcal{L}$ with signature σ. If I is a structure of signature σ, or *σ-structure*, we define the *$\mathcal{L}$ theory of I* as follows:

$$Th_{\mathcal{L}}(I) = \{\varphi \in \mathcal{L}_\sigma : I \models_{\mathcal{L}} \varphi\}$$

A *database schema* will be regarded as a *relational signature*, and a *database instance* of some schema σ as a finite and relational *σ-structure*. If φ is a sentence in $\mathcal{L}_\sigma$, we define

$$MOD(\varphi) = \{I \in \mathcal{B}_\sigma : I \models \varphi\}$$

By $\varphi(x_1, \ldots, x_r)$ we denote a formula of some logic whose free variables are *exactly* $\{x_1, \ldots, x_r\}$. Let $free(\varphi)$ be the set of free variables of the formula φ. If $\varphi(x_1, \ldots, x_k) \in \mathcal{L}_\sigma$, $I \in \mathcal{B}_\sigma$, $\bar{a}_k = (a_1, \ldots, a_k)$ is a k-tuple over I, let $I \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]$ denote that φ is TRUE, when interpreted by I, under a valuation v where for $1 \leq i \leq k$ $v(x_i) = a_i$. Then we consider the set of all such valuations as follows:

$$\varphi^I = \{(a_1, \ldots, a_k) : a_1, \ldots, a_k \in dom(I) \wedge I \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]\}$$

That is, φ^I is the relation defined by φ in the structure I, and its arity is given by the number of free variables in φ. Sometimes, we will use the same notation when the set of free variables of the formula is *strictly* included in $\{x_1, \ldots, x_k\}$. Formally, we say that a formula $\varphi(x_1, \ldots, x_k)$ of signature σ, *expresses* a query q of schema σ, if for every database I of schema σ, $q(I) = \varphi^I$. Similarly, a sentence φ expresses a Boolean query q if for every database I of schema σ, is $q(I) = 1$ iff $I \models \varphi$. We will also deal with *extensions* of structures. If R is a relation of arity k in the domain of a structure I, we denote as $\langle I, R\rangle$ the τ-structure resulting by adding the relation R to I, where τ is obtained from σ by adding a relation symbol of arity k. Similarly, if $\bar{a}_k$ is a k-tuple over I, we denote by $\langle I, \bar{a}_k\rangle$ the τ-structure resulting by adding the k-tuple $\bar{a}_k$ to I, where τ is obtained from σ by adding k constant symbols $c_1, \ldots, c_k$, and where for $1 \leq i \leq k$, the constant symbol c_i of τ is interpreted in I by the i-th component of $\bar{a}_k$. This is the only case where we allow

constant symbols in a signature. We denote by FO^k, where $k \geq 1$ is an integer, the fragment of FO where only formulas whose variables, either free or bound, are in $\{x_1, \ldots, x_k\}$ are allowed. In this setting, FO^k itself is a logic. This logic is obviously *less expressive* than FO. We denote as C^k the logic which is obtained by adding to FO^k *counting quantifiers*, i.e., all existential quantifiers of the form $\exists^{\geq m} x$ with $m \geq 1$. Informally, $\exists^{\geq m} x(\varphi)$ means that there are at least m *different* elements of the database which satisfy φ.

2.3 Types

Given a database I and a k-tuple $\bar{a}_k$ over I, we would like to consider *all* properties of $\bar{a}_k$ in the database I including the properties of every component of the tuple and the properties of all different sub-tuples of $\bar{a}_k$. Therefore, we use the notion of *type*. Let $\mathcal{L}$ be a logic. Let I be a database of some schema σ and let $\bar{a}_k = (a_1, \ldots, a_k)$ be a k-tuple over I. The $\mathcal{L}$ *type* of $\bar{a}_k$ in I, denoted $tp_I^{\mathcal{L}}(\bar{a}_k)$, is the set of formulas in $\mathcal{L}_\sigma$ with free variables *among* $\{x_1, \ldots, x_k\}$ such that every formula in the set is TRUE when interpreted by I for any valuation which assigns the i-th component of $\bar{a}_k$ to the variable x_i, for every $1 \leq i \leq k$. In symbols

$$tp_I^{\mathcal{L}}(\bar{a}_k) = \{\varphi \in \mathcal{L}_\sigma : free(\varphi) \subseteq \{x_1, \ldots, x_k\} \wedge I \models \varphi[a_1, \ldots, a_k]\}$$

It is noteworthy that, according to this definition, the $\mathcal{L}$ *theory* of the database I, i.e., $Th_{\mathcal{L}}(I)$, is included in the $\mathcal{L}$-type of *every* tuple over I. That is, the class of all the properties of a given tuple, which are expressible in $\mathcal{L}$, includes not only the properties of all its sub-tuples, but also the properties of the database itself.

Note that the type of two different k-tuples of the same database may be different, even if the types of their components are the same. Think of a complete binary tree of depth h. If we consider types for single elements (i.e., $k = 1$), then we have only $h + 1$ different types, because all the nodes of the same depth satisfy the same properties. They can be exchanged by an automorphism of the tree. Now let us consider types for pairs ($k = 2$). We can build two pairs, such that the type of the two first components is the same, and the type of the two second components is also the same, but the types of the two pairs are different. Just take for one pair a node in some depth > 0 and one of its sons, and for the other pair we take a node of the same depth as the first component of the first pair, and another node in the next level of the tree, but which is *not* the son of the first component.

We may also regard an $\mathcal{L}$-type as a *set of queries*, and even as a query. We can think of a type *without having a particular database in mind.* That is, we add properties (formulas with the appropiate free variables) as long as the resulting set remains consistent. Let us denote as $Tp^{\mathcal{L}}(\sigma, k)$ for some $k \geq 1$ the class of *all* $\mathcal{L}$-types for k-tuples over databases of schema σ. In symbols

$$Tp^{\mathcal{L}}(\sigma, k) = \{tp_I^{\mathcal{L}}(\bar{a}_k) : I \in \mathcal{B}_\sigma \wedge \bar{a}_k \in (dom(I))^k\}$$

Hence, $Tp^{\mathcal{L}}(\sigma, k)$ is a class of properties, or a set of sets of formulas. Let $\alpha \in Tp^{\mathcal{L}}(\sigma, k)$ (i.e., α is the $\mathcal{L}$-type of some k-tuple over some database in $\mathcal{B}_\sigma$). We

say that a database I *realizes* the type α if there is a k-tuple $\bar{a}_k$ over I whose $\mathcal{L}$-type is α. That is, if $tp_I^{\mathcal{L}}(\bar{a}_k) = \alpha$. We denote by $Tp^{\mathcal{L}}(I,k)$ the class of all $\mathcal{L}$-types for k-tuples which are realized in I. That is, it is the class of properties of all the k-tuples over the database I which can be expressed in $\mathcal{L}$. In symbols

$$Tp^{\mathcal{L}}(I,k) = \{tp_I^{\mathcal{L}}(\bar{a}_k) : \bar{a}_k \in (dom(I))^k\}$$

The following well known fact (see [Ott97]) relates types of tuples with C^k-theories of databases (and also FO, see Fact 2).

Fact 1. For every $k > 0$, for every schema σ, and for every pair of databases I, J of schema σ, the following holds:

$$I \equiv_{C^k} J \iff Tp^{C^k}(I,k) = Tp^{C^k}(J,k)$$

□

The following fact means that when two databases realize the same C^k-types for tuples, it doesn't matter which length of tuples we consider. It is well known (see [Ott97]).

Fact 2.

- For every $k > 0$, for every schema σ, for every pair of databases I, J of schema σ, and for every $1 \leq l \leq k$, the following holds:

$$Tp^{C^k}(I,k) = Tp^{C^k}(J,k) \iff Tp^{C^k}(I,l) = Tp^{C^k}(J,l)$$

- For every schema σ, for every pair of databases I, J of schema σ, and for every $l \geq 1$, the following holds:

$$I \equiv_{FO} J \iff Tp^{FO}(I,l) = Tp^{FO}(J,l)$$

□

Note that the $\mathcal{L}$-type of the 0-tuple is the $\mathcal{L}$-theory of the database.

The following is a well known result which, among other sources, can be found as Proposition 2.1.1 in [EF99].

Proposition 3. *For every schema σ and for every pair of (finite) databases I, J of schema σ the following holds:*

$$I \equiv_{FO} J \iff I \simeq J$$

□

Although types are infinite sets of formulas, a *single* C^k formula is equivalent to the C^k-type of a tuple over a given database. The equivalence holds for all databases of the same schema. This result has been proved by M. Otto.

Proposition 4. *([Ott96]): For every schema σ, for every database I of schema σ, for every $k \geq 1$, for every $1 \leq l \leq k$, and for every l-tuple $\bar{a}_l$ over I, there is a C^k formula $\chi \in tp_I^{C^k}(\bar{a}_l)$ such that for any database J of schema σ and for every l-tuple $\bar{b}_l$ over J*

$$J \models \chi[\bar{b}_l] \Longleftrightarrow tp_I^{C^k}(\bar{a}_l) = tp_J^{C^k}(\bar{b}_l)$$

□

Moreover, such a formula χ can be built inductively for a given database. If a C^k formula χ satisfies the condition of Proposition 4, we call χ an *isolating formula* for $tp_I^{C^k}(\bar{a}_l)$.

Concerning logical characterizations of databases, the next two propositions are quite important and well known, and we will make use of them in the present work (see [Ott97] and [EF99]). Recall that we are considering only *finite* databases in the present article.

Proposition 5. *Let $\mathcal{L} \in \{FO^k, C^k\}$, for some $k \geq 1$. Let I be a database of some schema σ. Then there exists a sentence α_I in $\mathcal{L}$ which characterizes I up to $\equiv_{\mathcal{L}}$, i.e., for every database J in $\mathcal{B}_\sigma$, the following holds:*

$$J \models \alpha_I \Longleftrightarrow I \equiv_{\mathcal{L}} J$$

□

Proposition 6. *Let $k \geq 1$. Then the following holds:*

(i) $\equiv_{FO^k}$ coincides with $\equiv_{\mathcal{L}^k_{\infty\omega}}$, i.e., for every schema σ, and for every two databases I, J in $\mathcal{B}_\sigma$:

$$I \equiv_{FO^k} J \Longleftrightarrow I \equiv_{\mathcal{L}^k_{\infty\omega}} J$$

(ii) $\equiv_{C^k}$ coincides with $\equiv_{\mathcal{C}^k_{\infty\omega}}$, i.e., for every schema σ, and for every two databases I, J in $\mathcal{B}_\sigma$:

$$I \equiv_{C^k} J \Longleftrightarrow I \equiv_{\mathcal{C}^k_{\infty\omega}} J$$

□

Let $\bar{a}_k = (a_1, \ldots, a_k)$ be a k-tuple over I. We say that the type $tp_I^{\mathcal{L}}(\bar{a}_k)$ is an *automorphism type* in the database I if for every k-tuple $\bar{b}_k = (b_1, \ldots, b_k)$ over I, if $tp_I^{\mathcal{L}}(\bar{a}_k) = tp_I^{\mathcal{L}}(\bar{b}_k)$, then there exists an automorphism f of the database I which maps $\bar{a}_k$ onto $\bar{b}_k$, i.e., for $1 \leq i \leq k$, $f(a_i) = b_i$. Regarding the tuple $\bar{a}_k$ in the database I, the logic $\mathcal{L}$ is therefore sufficiently expressive with respect to the properties which might make $\bar{a}_k$ distinguishable from other k-tuples in the database I. We say that the type $tp_I^{\mathcal{L}}(\bar{a}_k)$ is an *isomorphism type* if for every database $J \in \mathcal{B}_\sigma$, and for every k-tuple $\bar{b}_k = (b_1, \ldots, b_k)$ over J, if $tp_I^{\mathcal{L}}(\bar{a}_k) = tp_J^{\mathcal{L}}(\bar{b}_k)$, then there exists an isomorphism $f : dom(I) \to dom(J)$ which maps $\bar{a}_k$ in I onto $\bar{b}_k$ in J, i.e., for $1 \leq i \leq k$, $f(a_i) = b_i$.

2.4 Asymptotic Probabilities

See [EF99], among other sources. Let φ be a sentence in $\mathcal{L}_\sigma$. We define

$$MOD_n(\varphi) = \{I \in \mathcal{B}_\sigma : dom(I) = \{1, \ldots, n\} \wedge I \models \varphi\}$$

Let's denote as $\mathcal{B}_{\sigma,n}$ the sub-class of databases of schema σ with domain $\{1, \ldots, n\}$. We define the following limit, which we call *asymptotic probability of* φ:

$$\mu_\varphi = \lim_{n \to \infty} (|MOD_n(\varphi)|/|\mathcal{B}_{\sigma,n}|)$$

We say that a logic $\mathcal{L}$ has a 0–1 *Law* if for every sentence $\varphi \in \mathcal{L}$ μ_φ exists, and is either 0 or 1. The same notion can also be defined on classes of databases, or Boolean queries. This means that the asymptotic probability of every property which can be expressed in the formalism (or of the given class) always exists, and is either 0 or 1. Among other logics, FO has this Law.

3 Definition of the Hierarchy

One of the main reasons for the weakness of FO^k regarding expressibility of queries, is its inability to count beyond the bound given by k. For instance, note that we need $k+2$ different variables to express that a node in a graph has out degree exactly k. Hence, it seems quite natural to add to FO^k the capability to count beyond that bound, while still restricting the number of different variables which may be used in a formula. In this way we get the logic C^k (see Section 2), which turns out to be much more expressive than FO^k (see [EF99] and [Imm99]). In logics with counting, 2 variables are enough to express *any* out degree of a node in a graph. Then, we define in this section a hierarchy which is similar to the one defined in [Tur01a, Tur04], but for which we consider the logics C^k instead of FO^k. In this way we get a new hierarchy whose layers are much bigger than the corresponding layers in the hierarchy of [Tur01a, Tur04]. In Section 5 we compare the two hierarchies.

Definition 7. *Let σ be a database schema and let $k \geq 1$ and $k \geq r \geq 0$. Then we define*

$$\mathcal{QCQ}_\sigma^{C^k} = \{f^r \in \mathcal{CQ}_\sigma \mid \forall I, J \in \mathcal{B}_\sigma :$$

$$Tp^{C^k}(I,k) = Tp^{C^k}(J,k) \Longrightarrow Tp^{C^k}(\langle I, f(I)\rangle, k) = Tp^{C^k}(\langle J, f(J)\rangle, k)\}$$

where $\langle I, f(I)\rangle$ and $\langle J, f(J)\rangle$ are databases of schema $\sigma \cup \{R\}$, with R being a relation symbol of arity r.

We also require that the answer to the query f must be the union of complete C^k-types, i.e., for all databases I in $\mathcal{B}_\sigma$, and for all tuples $\bar{a}, \bar{b}$ in $dom(I)^r$, if $\bar{a} \in f(I)$ and $tp_I^{C^k}(\bar{a}) = tp_I^{C^k}(\bar{b})$, then also $\bar{b} \in f(I)$.

We define further $\mathcal{QCQ}^{C^k} = \bigcup_\sigma \mathcal{QCQ}_\sigma^{C^k}$ and $\mathcal{QCQ}^{C^\omega} = \bigcup_{k \geq 1} \mathcal{QCQ}^{C^k}$.

That is, a query is in the sub-class $\mathcal{QCQ}^{C^k}$ if it *preserves realization* of C^k-types for k-tuples. By preservation of C^k-types realization, we mean the property that for every pair of databases of the corresponding schema, say σ, and for every C^k-type for k-tuples (i.e., for every type in $Tp^{C^k}(\sigma, k)$), if both databases have the *same number* of k-tuples of that type then the relations defined by the query in each database also agree in the same sense, considering the schema of the databases with the addition of a relation symbol with the arity of the query. Note the difference with the classes $\mathcal{QCQ}^k$ in ([Tur01a], [Tur04]), where the cardinalities of the corresponding sets of tuples may be different. By Fact 1 equality in the set of C^k-types for k-tuples realized in two given databases is equivalent to $\equiv_{C^k}$, i.e., equality of C^k theories. Moreover, by Fact 2 the size of the tuples which we consider for the types is irrelevant. Thus, queries in $\mathcal{QCQ}^{C^k}$ may also be regarded as those which *preserve equality of C^k theories.* Note that the different classes $\mathcal{QCQ}^{C^k}$ form a hierarchy, i.e., for every $k \geq 1$, $\mathcal{QCQ}^{C^k} \subseteq \mathcal{QCQ}^{C^{k+1}}$. This follows from the notion of C^k-type and from the definition of the classes $\mathcal{QCQ}^{C^k}$. It can be also obtained as a straightforward corollary of Theorem 13.

Hence, queries in $\mathcal{CQ}$ may be considered as ranging from those for whose computation we need to consider every property of the input database up to isomorphism (i.e., every FO property), to those for whose computation it is enough to consider the C^k properties of the input database, for some fixed k. Different sub-classes $\mathcal{QCQ}^{C^k}$ in the hierarchy $\mathcal{QCQ}^{C^\omega}$ correspond to different degrees of "precision" with which we *need* to consider the input database to evaluate the queries in the sub-class on that database.

Next, we give an important result from [CFI92] which we will use in most of our proofs. Then, we show that the hierarchy defined by the sub-classes $\mathcal{QCQ}_\sigma^{C^k}$, for $k \geq 1$, is *strict.*

Proposition 8. *([CFI92]) For every $k \geq 1$, there are two non isomorphic graphs G_k, H_k, such that $G_k \equiv_{C^k} H_k$.* □

Proposition 9. *For every $k \geq 1$, there is some $h > k$ such that $\mathcal{QCQ}_\sigma^{C^h} \supset \mathcal{QCQ}_\sigma^{C^k}$.*

Proof. The inclusion is trivial and can also be easily obtained as a corollary to Theorem 13. For the strict inclusion we will use the graphs G_k, H_k of Proposition 8. Note that by Proposition 3, for every pair of the graphs G_k, H_k there exists an integer $h > k$ such that the C^h-types are FO-types for both graphs. Let us write h as $h(k)$. Then, for every $k \geq 1$, by Proposition 8 there are nodes in one of the graphs, say G_k, whose $C^{h(k)}$-types are not realized in the other graph H_k. Then we define for every $k \geq 1$, the query f_k in the schema of the graphs, say σ, as the nodes of the input graph whose $C^{h(k)}$-types are not realized in H_k. We will show first that $f_k \in \mathcal{QCQ}_\sigma^{C^{h(k)}}$. Let I, J be an arbitrary pair of graphs with $I \equiv_{C^{h(k)}} J$. If they are $C^{h(k)}$ equivalent to H_k, then the result of f_k will be the empty set for both graphs. If they are not $C^{h(k)}$ equivalent to H_k, then clearly the nodes in the result of f_k will have the same $C^{h(k)}$-types in both graphs. So, f_k preserves realization of $C^{h(k)}$-types and hence $f_k \in \mathcal{QCQ}_\sigma^{C^{h(k)}}$. Now, we will show that $f_k \notin \mathcal{QCQ}_\sigma^{C^k}$. To

see that, note that $f_k(H_k) = \emptyset$ by definition of f_k, but by Proposition 8 and by our assumption $f_k(G_k) \neq \emptyset$ and $H_k \equiv_{C^k} G_k$. Thus, f_k does not preserve realization of C^k-types and hence $f_k \notin \mathcal{QCQ}_\sigma^{C^k}$. □

Proposition 10. $\mathcal{QCQ}^{C^\omega} \subset \mathcal{CQ}$.

Proof. The inclusion is trivial, since clearly every query in $\mathcal{QCQ}^{C^\omega}$ is computable. For the strict inclusion we will use again the graphs G_k, H_k of Proposition 8. We define a query f on the schema of the pairs of disjoint graphs, say σ, as the nodes in the first graph, whose FO-type is not realized in the second graph. Clearly, f is computable and total. Now, towards a contradiction, let us assume that $f \in \mathcal{QCQ}^{C^\omega}$. Then, for some $h \geq 1$, $f \in \mathcal{QCQ}_\sigma^{C^h}$. For some order of the pairs of corresponding graphs as in Proposition 8, say (G_h, H_h), the result of f is non empty, by the definition of f. If we consider now the pair (G_h, G_h), the result of f is empty. Since $(G_h, H_h) \equiv_{C^h} (G_h, G_h)$, it turns out that $f \notin \mathcal{QCQ}_\sigma^{C^h}$, which is a contradiction. Thus, $f \notin \mathcal{QCQ}^{C^\omega}$. □

3.1 A Reflective Machine for Logics with Counting.

We will now define a model of computation to characterize the sub-classes $\mathcal{QCQ}_\sigma^{C^k}$.

In [Ott96], M. Otto defined a new model of computation of queries inspired by the *RM* of [AV95], to characterize the expressive power of fixed point logic with *counting terms* (see [Imm99]). Here, we define a machine which is similar to the model of Otto, but which is inspired by the *RRM* of [APV98], instead. In this paper, we will not compare the expressive power of the model of Otto and ours. However, it is straightforward to prove that the machine of Otto can be simulated in our model.

Definition 11. *For every $k \geq 1$, we define the* reflective counting machine of variable complexity k *which we denote by RCM^k, as a reflective relational machine RRM^k where dynamic queries are C^k formulas, instead of FO^k formulas. In all other aspects, our model works in exactly the same way as RRM^k. We define $RCM^{O(1)} = \bigcup_{k \geq 1} RCM^k$.*

We need first a technical result. Then, we can prove the characterization of the expressive power of the model RCM^k.

Let φ be a formula of some logic $\mathcal{L}$, where the relation symbols $R_1, \ldots, R_k$, with arities $r_1, \ldots, r_k$, are used. Let $\varphi_1(x_{11}, \ldots, x_{1r_1}), \ldots, \varphi_k(x_{k1}, \ldots, x_{kr_k})$ be also formulas in $\mathcal{L}$. We say that $\hat{\varphi}$ *is obtained from φ by composition with* $\varphi_1, \ldots, \varphi_k$, if for every $1 \leq i \leq k$, every occurrence of an atomic formula of the form $R_i(z_1, \ldots, z_{r_i})$ in φ, is replaced by the formula φ_i in such a way that, for all $1 \leq j \leq r_i$, each occurrence of the free variable x_{ij} in φ_i is replaced by the variable z_j.

Lemma 12. *Let $k \geq 1$, let σ be a schema, let I be a database of schema σ and let $\mathcal{M}$ be an RCM^k which computes a query of schema σ. Then, there is a formula $\varphi_{\mathcal{M},I}$ in C^k which defines on I the relation resulting from the computation of $\mathcal{M}$ on input I. Moreover, $\varphi_{\mathcal{M},I}$ depends only on $\mathcal{M}$ and I.*

Proof. The only transitions of $\mathcal{M}$ which may affect the content of the output relation in the relational store of $\mathcal{M}$, are those transitions where a C^k formula is evaluated, and the relation defined by the evaluation is assigned to some relation symbol in the relational store. Informally, in every computation step where a C^k formula, say ψ, is evaluated, we can use composition as defined above, replacing each relation symbol R which is used in ψ, and which is not in σ, by the last C^k formula whose resulting relation from its evaluation on the relational store was assigned to R. Then, by induction on the length of the computation of $\mathcal{M}$ on input I, and by applying composition, it is straightforward to prove that we get finally a C^k formula $\varphi_{\mathcal{M},I}$ which, evaluated on I, defines the same output relation as the computation of $\mathcal{M}$ on input I. And clearly this formula depends only on $\mathcal{M}$ and I. Finally, note that C^k is closed under composition, as defined above. □

Theorem 13. *For every $k \geq 1$, the class of total queries which are computable by RCM^k machines is exactly the class $\mathcal{QCQ}^{C^k}$.*

Proof. **a)** ($\subseteq$): Suppose that an r-ary query f of schema σ is computable by a RCM^k $\mathcal{M}$, for some $k \geq r$. And let I and J be two databases of schema σ such that $Tp^{C^k}(I,k) = Tp^{C^k}(J,k)$. According to the definition of the RCM^k machine, the only way to assign a relation to an r-ary relation symbol in the relational store is through a C^k formula with r free variables, say φ. And, by Fact 18 the result of the evaluation of φ on the database in the relational store of $\mathcal{M}$ is the projection of the union of some equivalence classes in the relation of equality of C^k-types for k-tuples over I. That is, φ is equivalent to the disjunction of some isolating formulas of C^k-types for r-tuples. But, by definition, the isolating formulas express the same C^k-types in every database of the corresponding schema. Thus, the r-tuples from $dom(J)$ which form the relation induced by φ in J must be those with the same C^k-types as the r-tuples from $dom(I)$ which form the relation induced by φ in I. So $Tp^{C^k}(\langle I, f(I)\rangle, r) = Tp^{C^k}(\langle J, f(J)\rangle, r)$. By Fact 2, also $Tp^{C^k}(\langle I, f(I)\rangle, k) = Tp^{C^k}(\langle J, f(J)\rangle, k)$, so $f \in \mathcal{QCQ}^{C^k}$. Note that the only way for the machine not to evaluate the same formula φ for both databases is the existence of a C^k sentence which evaluates to different truth values on I and J. But this is not possible by Fact 1 because $I \equiv_{C^k} J$.

b) ($\supseteq$): Let $f \in \mathcal{QCQ}^{C^k}$ be an r-ary query of schema σ for some $k \geq r$. We build an RCM^k machine $\mathcal{M}_f$, which will compute f. We use a countably infinite number of k-ary relation symbols in its relational store. With the input database I in its relational store, $\mathcal{M}_f$ will build an encoding of a database I' in its TM tape such that $Tp^{C^k}(I,k) = Tp^{C^k}(I',k)$. For this purpose, $\mathcal{M}_f$ will work as follows:

(i) $\mathcal{M}_f$ finds out the size of I, say n. Note that this can be done through an iteration by varying m in the query $\exists^{\geq m} x(x = x) \wedge \neg\exists^{\geq m+1} x(x = x)$ which is in C^1.

(ii) Then $\mathcal{M}_f$ builds an encoding of every possible database I' of schema σ and of size n in its TM tape with domain $\{1, \ldots, n\}$.

(iii) For every I' and for every k-tuple s_i over I', $\mathcal{M}_f$ builds on its TM tape the isolating formula χ_{s_i} (as in Proposition 4) for the C^k-type of s_i in the database I', and in this way we get isolating formulas for the types in $Tp^{C^k}(I', k)$.

(iv) $\mathcal{M}_f$ evaluates as dynamic queries the formulas χ_{s_i}, for every i, which are in C^k and assigns the results to the working k-ary relation symbols S_i in the relational store, respectively (note that these queries are evaluated on the input database I).

(v) If every relation S_i is non-empty, and if the union of all of them is the set of k-tuples over I, then it means that $Tp^{C^k}(I, k) = Tp^{C^k}(I', k)$ and I' is the database we were looking for; otherwise, we try another database I'. Note that $\mathcal{M}_f$ can check whether the relation S_i is empty with the dynamic query $\exists x_1 \dots x_k(S_i(x_1, \dots, x_k))$.

(vi) Now $\mathcal{M}_f$ computes $f(I')$ on its TM tape, which is possible because $f \in \mathcal{CQ}$ and f is defined on I' because it is total, and then expands the r-ary relation $f(I')$ to a k-ary relation $f^k(I')$ by taking the cartesian product with $dom(I')$.

(vii) $\mathcal{M}_f$ builds $f^k(I)$ in the relational store as the union of the relations S_i which correspond to the C^k-types χ_{s_i} of the k-tuples s_i which form $f^k(I')$, and finally it reduces the k-ary relation $f^k(I)$ to an r-ary relation $f(I)$.

□

Corollary 14. $RCM^{O(1)} = \mathcal{CQ}^{C^\omega}$. □

Corollary 15. *The computation model $RCM^{O(1)}$ is incomplete. That is, there are computable queries which cannot be computed by any $RCM^{O(1)}$ machine.* □

Corollary 16. *Let $f \in \mathcal{CQ}$, of some schema σ. Then, for every $k \geq 1$ and for every natural number i, if f preserves realization of C^k-types for k-tuples for every pair of databases in $\mathcal{B}_\sigma$, then f also preserves realization of C^{k+i}-types for $(k+i)$-tuples for every pair of databases in $\mathcal{B}_\sigma$.* □

Remark 17. The hierarchy defined by the classes $\mathcal{CQ}^{C^k}$ is not only *strict*, but it is *orthogonal* to the hierarchy of complexity classes defined in terms of TIME and SPACE of Turing machines (like $LOGSPACE \subseteq PTIME \subseteq NP \subseteq PSPACE \subseteq EXPTIME$). This is also the case with the classes $\mathcal{CQ}^k$ of ([Tur01a], [Tur04]). Note that *any* recursive predicate, evaluated on the number of equivalence classes in the (equivalence) relation defined by equality of C^k-types in the set of k-tuples of a database, is in $\mathcal{CQ}^{C^k}$. Therefore, there is no complexity class defined by any bounds in TIME or SPACE of Turing machines which may include $\mathcal{CQ}^{C^k}$. And this is the case for every $k \geq 1$. In Section 5 we make some considerations to this regard.

4 Homogeneity in Logics with Counting

In this section, we will study some properties of databases which are relevant to the computation power of the machines $RCM^{O(1)}$, when working on databases with such properties. That is, we will prove that there are important queries which cannot be computed by an $RCM^{O(1)}$ on the whole class of databases of a given schema, but whose restriction to certain classes of databases are computable by such machines.

First, we will give some definitions and well known facts which we need (see [Ott97]). Let k, l be positive integers such that $k \geq l \geq 1$. Let us denote by $\equiv_k$ the (equivalence) relation induced on the set of l-tuples over a given database I by equality of C^k-types of l-tuples. That is, for every pair of l-tuples $\bar{a}_l$ and $\bar{b}_l$ over I, $\bar{a}_l \equiv_k \bar{b}_l$ iff $tp_I^{C^k}(\bar{a}_l) = tp_I^{C^k}(\bar{b}_l)$.

Fact 18. For every schema σ, for every database I of schema σ, for every $k \geq 1$, for every $1 \leq r \leq k$, and for every C^k formula φ of signature σ, with r free variables, the relation which φ defines on I is the projection of the union of some equivalence classes in the relation $\equiv_k$ for k-tuples over I. □

When the query which φ expresses is of arity k, i.e., when $r = k$, the result simply means that the query cannot distinguish between two k-tuples whose C^k-types are the same. On the other hand, if $r < k$, the intuitive idea is the same, but of course the k-tuples in the equivalence classes of $\equiv_k$ must be reduced to r-tuples in some uniform way.

So that the equivalence classes in the relation $\equiv_k$ for r-tuples, are unions of equivalence classes in the relation $\equiv_k$ for k-tuples, reduced to r-tuples in some uniform way. And this is what the first part of the following fact shows.

Fact 19.

- For every schema σ, for every database I of schema σ, for every $k \geq 1$, for every $1 \leq r \leq k$, every equivalence class in the relation $\equiv_k$ for r-tuples over I, is the projection of the union of some equivalence classes in the relation $\equiv_k$ for k-tuples over I.
- For every schema σ, for every database I of schema σ, for every $k \geq 1$, for every $1 \leq r \leq k$, every equivalence class in the relation $\equiv_r$ for r-tuples over I, is the projection of the union of some equivalence classes in the relation $\equiv_k$ for k-tuples over I. □

From Fact 19 it follows that allowing more variables in isolating formulas for C^k-types of tuples results in a more precise view of the properties which identify a given sub-set of tuples among the whole set of tuples which may be built over the database. By Proposition 3, we know that the limit is FO which includes the logic C^k for every natural k. Types in FO are isomorphism types (and, hence, also automorphism types) for tuples of every length in every database. So, we want to consider the number of variables which are needed for a given database and for a

given integer k to express in FO with counting quantifiers the properties of the k-tuples in that database up to automorphism. For this purpose we define different variants of the notion of homogeneity from model theory (see [EF99] and [CK92]) in the context of logics with counting. For every $k \geq 1$ and for any two k-tuples $\bar{a}_k$ and $\bar{b}_k$ over a given database I, we will denote as $\equiv_\simeq$ the (equivalence) relation defined by the existence of an automorphism in the database I mapping one k-tuple onto the other. That is, $\bar{a}_k \equiv_\simeq \bar{b}_k$ iff there exists an automorphism f on I such that, for every $1 \leq i \leq k$, $f(a_i) = b_i$.

Let $k \geq 1$. A database I is *C^k-homogeneous* if for every pair of k-tuples $\bar{a}_k$ and $\bar{b}_k$ over I, if $\bar{a}_k \equiv_k \bar{b}_k$, then $\bar{a}_k \equiv_\simeq \bar{b}_k$. Let σ be a schema. A class $\mathcal{C}$ of databases of schema σ is *C^k-homogeneous* if every database $I \in \mathcal{C}$ is C^k-homogeneous.

The next fact is immediate (see [Ott97]).

Fact 20. Let $k \geq 1$. If a database I is C^k-homogeneous, then for every $1 \leq l \leq k$ and for every pair of l-tuples $\bar{a}_l$ and $\bar{b}_l$ over I, if $\bar{a}_l \equiv_k \bar{b}_l$, then $\bar{a}_l \equiv_\simeq \bar{b}_l$. □

We define next a presumably stronger notion regarding homogeneity: *strong C^k-homogeneity.* To the author's knowledge it is not known whether there exist examples of classes of databases which are C^k-homogeneous for some k, but which are not strongly C^k-homogeneous. This was also the case with the analogous notions in ([Tur01a], [Tur04]). However, the consideration of strong C^k-homogeneity makes sense not only because of the intuitive appeal of the notion, but also because this is the property which we use in Proposition 27 to prove that the class $\mathcal{QCQ}^C$ (see Definition 26) is a lower bound with respect to the increment in computation power of the machine RCM^k when working on strongly C^k-homogeneous databases. Up to know, we could not prove that this result holds for C^k-homogeneous databases as well. Let $k \geq 1$. A database I is *strongly C^k-homogeneous* if it is C^r-homogeneous for every $r \geq k$. Let σ be a schema. A class $\mathcal{C}$ of databases of schema σ is *strongly C^k-homogeneous* if every database $I \in \mathcal{C}$ is strongly C^k-homogeneous.

Note that, by Proposition 3, every database is strongly C^k-homogeneous for some $k \geq 1$. However, it is clear that this is not the case with *classes* of databases.

In [Ott96] it was noted that the discerning power of the machine defined there, which is similar to our machine RCM^k (see discussion before Definition 11), is restricted to C^k-types for k-tuples. So, the following result seems quite natural, and is somehow implicit in Otto's work. If f is a query of schema σ, and $\mathcal{C}$ is a class of databases of schema σ, we will denote as $f|_\mathcal{C}$ the restriction of f to the class $\mathcal{C}$.

Theorem 21. *For every schema σ and for every $k \geq 1$, there is a query f such that if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of databases of schema σ such that $\mathcal{C}$ is C^k-homogeneous and $\mathcal{C}'$ is not C^k-homogeneous, then $f|_\mathcal{C}$ is computable by an RCM^k machine but $f|_{\mathcal{C}'}$ is not computable by any RCM^k machine.*

Proof. For every $k \geq 1$, and for every database I, we define the Boolean query f as follows: $f(I) = TRUE$ *iff the number of equivalence classes in the relation* $\equiv_\simeq$ *for k-tuples over I is even.* The query f is clearly computable by an RCM^k machine on classes of databases which are C^k-homogeneous. To see this, note

that we can use the same construction as in the proof of Theorem 13 to build the isolating formulas for C^k-types for k-tuples, which for C^k-homogeneous classes of databases, by definition, are also automorphism types for k-tuples. Note that different relation symbols S_i of item *iv* in that proof can have the same contents, since there can be several k-tuples of the same C^k-types in I. For each pair S_i, S_j of those relations, $\mathcal{M}_f$ can check whether they are different simply with the query $\forall x_1 \ldots x_k (S_i(x_1, \ldots, x_k) \leftrightarrow S_j(x_1, \ldots, x_k))$. Then, $\mathcal{M}_f$ can count on its TM tape the number of different relations S_i. Note that this is the number of C^k-types for k-tuples realized in I, and as $I \in \mathcal{C}$ and $\mathcal{C}$ is C^k-homogeneous, this is also the number of equivalence classes in $\equiv_{\simeq}$.

On the other hand, for some $J \in \mathcal{C}'$, J is not C^k-homogeneous. Then in that database J, the number of C^k-types realized is different than the number of equivalence classes in $\equiv_{\simeq}$ since, by definition, there are at least two k-tuples $\bar{a}, \bar{b}$ in J such that $\bar{a} \equiv_k \bar{b}$ but $\bar{a} \not\equiv_{\simeq} \bar{b}$. And, as we saw in Theorem 13, no RCM^k machine can distinguish between the k-tuples $\bar{a}, \bar{b}$ in J, so that no such machine can count the number of equivalence classes in $\equiv_{\simeq}$ on that database. □

Remark 22. Note that the sub-class of queries whose restriction to C^k-homogeneous databases can be computed by RCM^k machines, but which *cannot* be computed by such machines on arbitrary classes of databases, is quite big. As we saw in the proof of the previous theorem, an RCM^k machine can count the number of equivalence classes in $\equiv_{\simeq}$ for k-tuples on a C^k-homogeneous database as an intermediate result on its TM tape. Then we can use this parameter, which we will call *type index for k-tuples* following [AV95], as the argument for *any* recursive predicate in an RCM^k machine. Thus, the *whole* class of recursive predicates, evaluated on the type index for k-tuples of the input database, is included in the sub-class of queries whose restriction to C^k-homogeneous databases can be computed with RCM^k machines, but which cannot be computed by such machines on arbitrary classes of databases.

However, we still do not know whether the machine RCM^k can achieve *completeness* when computing queries on databases which are C^k-homogeneous. This problem has important consequences in query computability and complexity, and is also related to the expressibility of fixed point logics (see [Ott96]). In particular, it is related to the problem of knowing whether whenever two databases are C^k-homogeneous, and are also C^k equivalent then they are isomorphic.

Next, we will show that the property of *strong* C^k-homogeneity will allow $RCM^{O(1)}$ machines to extend their power with respect to computability of queries.

Proposition 23. *For every schema σ and for every $k \geq 1$, there is a class of queries $F = \{f_r\}_{r \geq k}$ such that, if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of databases of schema σ such that $\mathcal{C}$ is strongly C^k-homogeneous and $\mathcal{C}'$ is not strongly C^k-homogeneous, then for every $f_r \in F$, $f_r|_{\mathcal{C}}$ is computable by an RCM^r machine, but it is* not *the case that for every $f_r \in F$, $f_r|_{\mathcal{C}'}$ is computable by an RCM^r machine.*

Proof. We can use here the same strategy as we used in the proof of Theorem 21. For every $r \geq k$, we define f_r as follows: $f_r(I) = TRUE$ *iff the number of*

equivalence classes in the relation $\equiv_\simeq$ for r-tuples over I is even. Clearly, for $I \in \mathcal{C}$, and for every $r \geq k$, the equivalence classes in $\equiv_r$ coincide with the equivalence classes in $\equiv_\simeq$ for r-tuples over I. So, for each $r \geq k$, f_r can be computed by an RCM^r machine, as we did in the proof of Theorem 21.

On the other hand, for some $J \in \mathcal{C}'$, which is not strongly C^k-homogeneous, there wil be some $r \geq k$, such that the equivalence classes in $\equiv_r$ will not coincide with the equivalence classes in $\equiv_\simeq$ for r-tuples over J. Then no RCM^r machine will be able to distinguish between r-tuples which have the same C^r-type, and hence no RCM^r machine will be able to count the equivalence classes in $\equiv_\simeq$ for r-tuples over J. □

With the next notion of homogeneity we intend to formalize the property of the classes of databases where C^k equivalence coincides with isomorphism. Among other classes, this is the case with the class of trees ([IL90]), the class of planar graphs ([Gro98b]) and the class of databases of bounded tree-width ([GM98]). With this stronger notion we can achieve *completeness* with RCM^k machines.

Definition 24. *Let σ be a schema and let $\mathcal{C}$ be a class of databases of schema σ. Let $k \geq 1$. We say that $\mathcal{C}$ is* pairwise C^k-homogeneous, *if for every pair of databases I, J in $\mathcal{C}$, and for every pair of k-tuples $\bar{a}_k \in (dom(I))^k$ and $\bar{b}_k \in (dom(J))^k$, if $\bar{a}_k \equiv_k \bar{b}_k$, then there exists an isomorphism $f : dom(I) \longrightarrow dom(J)$ such that $f(a_i) = b_i$ for every $1 \leq i \leq k$.*

Proposition 25. *For every schema σ, for every $k \geq 1$ and for every query f of schema σ in $\mathcal{CQ}$, if $\mathcal{C}$ is a recursive and pairwise C^k-homogeneous class of databases of schema σ, then there is an RCM^k machine which computes $f|_\mathcal{C}$.*

Proof. We use the same strategy to build an RCM^k machine $\mathcal{M}_f$ which computes a given query $f \in \mathcal{CQ}$, as we did in the proof of Theorem 13. In this case we must also check that the database I' which we build in the TM tape is in the class of databases on which $\mathcal{M}_f$ is defined to work. This is the reason why we ask for the class to be recursive. So we will know that if I' and I (the input database) realize the same C^k-types for k-tuples, then the two databases are isomorphic, and we can compute f on I' in the TM part of $\mathcal{M}_f$. Finally, to build $f(I)$, we just take the corresponding equivalence classes of the k-tuples which realize in I the C^k-types for k-tuples which are realized in $f(I')$. □

4.1 A Lower Bound for Strong Homogeneity

The queries which preserve realization of FO-types for tuples (i.e., the computable queries), but which do not preserve realization of C^k-types for k-tuples for any k, do not belong to *any* $\mathcal{QCQ}^{C^k}$ class. This is because if we fix some $k \geq 1$, the number of variables which are needed for the automorphism types for k-tuples in different databases may be different. And the number of different bounds for the number of variables may be infinite. Thus, we define a new class of queries, following the same strategy as in [Tur01a, Tur04] regarding FO^k. The intuitive idea behind

this new class is that queries which belong to it preserve, for any two databases of the corresponding schema, the realization of types in C^k where k is the number of variables which is *sufficient* for *both* databases to define tuples up to automorphism.

Definition 26. *For any schema σ, let us denote as $arity(\sigma)$ the maximum arity of a relation symbol in the schema. We define the class $\mathcal{QCQ}^C$ as the class of queries $f \in \mathcal{CQ}$ of some schema σ and of any arity, for which there exists an integer $n \geq max\{arity(f), arity(\sigma)\}$ such that for every pair of databases I, J in $\mathcal{B}_\sigma$, the following holds:*

$$Tp^{C^h}(I,h) = Tp^{C^h}(J,h) \Longrightarrow Tp^{C^h}(\langle I, f(I)\rangle, h) = Tp^{C^h}(\langle J, f(J)\rangle, h)$$

where $\langle I, f(I)\rangle$ and $\langle J, f(J)\rangle$ are databases of schema $\sigma \cup \{R\}$ with R being a relation symbol with the arity of the query f, and $h = max\{n, min\{k : I \text{ and } J \text{ are strongly } C^k\text{-homogeneous}\}\}$.

We also require that the answer to the query f must be the union of complete C^h-types, i.e., for all databases I in $\mathcal{B}_\sigma$, and for all tuples $\bar{a}, \bar{b}$ in $dom(I)^r$, if $\bar{a} \in f(I)$ and $tp_I^{C^h}(\bar{a}) = tp_I^{C^h}(\bar{b})$, then also $\bar{b} \in f(I)$.

Clearly, $\mathcal{QCQ}^C \supseteq \mathcal{QCQ}^{C^\omega}$. But, unlike the analogous class $\mathcal{QCQ}$ in [Tur01a, Tur04], we do not know neither whether the inclusion is strict, nor whether $\mathcal{CQ}$ strictly includes $\mathcal{QCQ}^C$. Both questions seem to be non trivial since they are related to the problem which we mentioned after Remark 22, by Proposition 27 below.

Note that the queries based on the C^k-type index for k-tuples which we mentioned in Remark 22 are in $\mathcal{QCQ}^C$. Therefore, the class of queries which can be computed by $RCM^{O(1)}$ machines on classes of databases which are C^k-homogeneous is actually quite big. It is big enough to allow for a RCM^k machine to go through the whole hierarchy $\mathcal{QCQ}^{C^\omega}$, in a sense which will be clarified next, and, further, to get into $\mathcal{QCQ}^C$. This is actually quite natural, because classes of databases which are strongly C^k-homogeneous will not benefit from the possibility of using $> k$ variables in dynamic queries, since C^k-types on them cannot be further refined, as long as the number of variables which may be used remains constant for the whole class of databases. So, the next result is rather intuitive.

Proposition 27. *Let f be a query of schema σ in $\mathcal{QCQ}^C$, with parameter n according to Definition 26. Let $\mathcal{C}$ be a class of databases of schema σ which is strongly C^k-homogeneous for some $k \geq 1$. Then, the restriction of f to $\mathcal{C}$ is computable by an RCM^h machine where $h = max\{n, k\}$.*

Proof. Let f and $\mathcal{C}$ be as in the statement of the proposition, and let f' be the restriction of f to $\mathcal{C}$. Let $I, J \in \mathcal{C}$, such that I is is strongly C^{k_1}-homogeneous and J is strongly C^{k_2}-homogeneous. Without loss of generality, let $k_1 \leq k_2 \leq k$. We must prove that f' can be computed on I and J by an RCM^n machine, if $k \leq n$, or by an RCM^k machine, otherwise. If $k \leq n$, by Definition 26, if $Tp^{C^n}(I,n) = Tp^{C^n}(J,n)$, then $Tp^{C^n}(\langle I, f'(I)\rangle, n) = Tp^{C^n}(\langle J, f'(J)\rangle, n)\}$. So, by Definition 7 and Theorem 13, f' can be computed on I and J by a RCM^n machine. As for every pair

of databases in $\mathcal{C}$, the minimum k' such that both databases are strongly $C^{k'}$-homogeneous is bounded by k, then f' preserves realization of C^n-types for n-tuples in $\mathcal{C}$. So, f' is computable by a RCM^n machine.

If, on the other hand, $k > n$, we have two different cases. Either $k_2 \leq n < k$ or $n \leq k_2 \leq k$. In the first case f' preserves realization of C^n-types for n-tuples in I and J, and by Corollary 16 f' also preserves realization of C^k-types for k-tuples in those databases. In the second case f' preserves realization of C^{k_2}-types for k_2-tuples in I and J, and by the same corollary f' also preserves realization of C^k-types for k-tuples in those databases. Then, by the same argument as before, in both cases f' preserves realization of C^k-types for k-tuples for any pair of databases in $\mathcal{C}$. So, f' is computable by a RCM^k machine. □

5 Further Considerations

5.1 Infinitary Logics with Counting

The *infinitary logic* $\mathcal{L}^k_{\infty\omega}$, for every $k \geq 1$, has the usual first order rules for the formation of formulas. In addition, it is closed under infinitary conjunctions and disjunctions. Formulas in $\mathcal{L}^k_{\infty\omega}$ contain at most k different variables. We write $\mathcal{L}^\omega_{\infty\omega} = \bigcup_k \mathcal{L}^k_{\infty\omega}$. Similarly, we define *infinitary logics with counting*, denoted as $C^k_{\infty\omega}$ and $C^\omega_{\infty\omega}$, respectively. These logics are defined in the same way as $\mathcal{L}^k_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ with the addition of all *counting quantifiers*. That is, $C^k_{\infty\omega}$ has the same formation rules as $\mathcal{L}^k_{\infty\omega}$ plus the following one: if ψ is a formula in $C^k_{\infty\omega}$, x is a variable and $m \geq 1$, then $\exists^{\geq m} x(\psi)$ is also a formula in $C^k_{\infty\omega}$, provided the number of different variables in $\exists^{\geq m} x(\psi)$ is $\leq k$. Then $C^\omega_{\infty\omega} = \bigcup_{k\geq 1} C^k_{\infty\omega}$.

We define the following recursive fragments of the infinitary logics $C^\omega_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ as in [AVV95]. For every $k \geq 1$, let $C^k_{\infty\omega}|_{rec}$ denote the fragment of the infinitary logic $C^k_{\infty\omega}$ that contains exactly all sentences with a recursive class of models. We also define $\mathcal{L}^k_{\infty\omega}|_{rec}$ in the same way. Now we give a characterization of the expressive power of the computation model RCM^k (and, hence, also of each subclass $\mathcal{QC}^{C^k}$) in terms of the fragment $C^k_{\infty\omega}|_{rec}$.

Theorem 28. *For every $k \geq 1$, the expressive power of the RCM^k machine, restricted to Boolean queries, is exactly $C^k_{\infty\omega}|_{rec}$.*

Proof. $\subseteq$): Let $\mathcal{M}$ be an RCM^k of schema σ for some natural k. By Propositions 5 and 6, every database I is characterized up to $\equiv_{C^k_{\infty\omega}}$ by a C^k formula. Let α_I be such a formula. On the other hand, by Lemma 12, the computation of $\mathcal{M}$ on a database I is equivalent to a C^k formula $\varphi_{\mathcal{M},I}$. Then we build the $C^k_{\infty\omega}$ formula $\Psi_{\mathcal{M}} \equiv \bigvee_{I\in\mathcal{B}_\sigma}(\alpha_I \wedge \varphi_{\mathcal{M},I})$. Clearly, the models of the formula $\Psi_{\mathcal{M}}$ are the databases accepted by $\mathcal{M}$. To see that the formula is in the fragment $C^k_{\infty\omega}|_{rec}$, note that we can build a Turing machine M which simulates $\mathcal{M}$ and which accepts a database iff it is accepted by $\mathcal{M}$.

$\supseteq$): Let Ψ be a $C^k_{\infty\omega}$ formula of schema σ such that its class of models is recursive, i.e., it is decidable by some Turing machine M. We build an RCM^k

machine $\mathcal{M}$ which works as follows. On input I it builds on the TM tape an encoding of every possible database I' of schema σ building the corresponding isolating formulas for the C^k-types realized in I', as in the proof of Theorem 13, until $I \equiv_{C^k} I'$. Since by Proposition 6 in finite databases C^k equivalence coincides with $C^k_{\infty\omega}$ equivalence, we have $I \models \Psi$ iff $I' \models \Psi$. Hence, $\mathcal{M}$ simulates the TM M in its tape on input I' and $\mathcal{M}$ accepts I iff M accepts I'. □

We can use the same strategy to characterize the expressive power of the model RRM^k (and, hence also of each subclass $\mathcal{QCQ}^k$ of ([Tur01a], [Tur04]) in terms of the corresponding fragment $\mathcal{L}^k_{\infty\omega}|_{rec}$. For the proof, we use Corollary 2.3 of [Ott97] and a result from ([APV98], Proof of Theorem 5.6) which is analogous to our Lemma 12 for RRM^k machines.

Theorem 29. *For every $k \geq 1$, the expressive power of the RRM^k machine, restricted to Boolean queries, is exactly $\mathcal{L}^k_{\infty\omega}|_{rec}$.* □

Now we will get a similar characterization for the classes of machines which might not halt on all input databases, i.e., for those machines which compute *partial* queries. For the rest of the present sub-section we will consider partial computable queries, that is, queries which are not necessarily defined on all the databases of a given schema. It is noteworthy that this is the way in which computable queries were originally defined (see [CH80] and [AHV94]), though usually only total queries are considered in the literature.

Following the definition in [AVV95] for the logic $\mathcal{L}^\omega_{\infty\omega}$, for every $k \geq 1$, let $C^k_{\infty\omega}|_{r.e.}$ denote the fragment of the infinitary logic $C^k_{\infty\omega}$ that contains exactly all sentences whose classes of models are recursively enumerable. We also define $\mathcal{L}^k_{\infty\omega}|_{r.e.}$ in the same way. Now we give a characterization of the expressive power of the computation models RCM^k when we consider also machines which compute partial queries, in terms of the fragment $C^k_{\infty\omega}|_{r.e.}$.

Theorem 30. *For every $k \geq 1$, the expressive power of the RCM^k machine which computes partial queries, restricted to Boolean queries, is exactly $C^k_{\infty\omega}|_{r.e.}$.*

Proof. $\subseteq$): We follow a similar strategy to the one used in the proof of Theorem 28. For the construction of the $C^k_{\infty\omega}$ formula $\Psi_{\mathcal{M}}$, we only consider the databases I for which the machine $\mathcal{M}$ halts, i.e., the databases on which the query computed by $\mathcal{M}$ is defined. The models of $\Psi_{\mathcal{M}}$ are clearly the databases for which the query computed by $\mathcal{M}$ evaluates to true. To prove that the class of models of $\Psi_{\mathcal{M}}$ is r.e., we construct a Turing machine M which builds on its work tape a sequence (which of course is countably infinite) of all the databases of the schema of the query of every possible size, ordered by their size. At the same time M simulates the operation of the RCM^k $\mathcal{M}$ on all the different databases built on the work tape, executing one step at a time on each of them, in such a way that after M executes the first step of the computation of $\mathcal{M}$ on a given database in the sequence, say I_i, it then executes one more step of the computation of $\mathcal{M}$ on each one of the previous databases in the sequence, i.e., on $I_1, I_2, \ldots, I_{i-1}$. Whenever the simulation of $\mathcal{M}$ on a given database halts, if it halts in an accepting state, then M outputs that

database in the sequence which it builds on the output tape of the machine. Then, M stops the simulation of $\mathcal{M}$ on that particular database. In this way we get a recursive enumeration of the models of the sentence $\Psi_{\mathcal{M}}$ in the output tape of M.

$\supseteq$): In this case we also follow a similar strategy as in the corresponding case in the proof of Theorem 28. Once $\mathcal{M}$ built on its TM tape an encoding of a database I' such that $I \equiv_{C^k} I'$, the machine $\mathcal{M}$ uses the TM M of the first part of the proof of the present theorem, to enumerate the models of $\Psi_{\mathcal{M}}$. Then, $\mathcal{M}$ accepts the input database if the database I' is generated in the enumeration of M. □

Using an analogous strategy we get a characterization of the expressive power of the computation models RRM^k when we consider also machines which compute partial queries, in terms of the fragment $\mathcal{L}^k_{\infty\omega}|_{r.e.}$.

Theorem 31. *For every $k \geq 1$, the expressive power of the RRM^k machines which compute partial queries, restricted to Boolean queries, is exactly $\mathcal{L}^k_{\infty\omega}|_{r.e.}$.* □

5.2 The Relevance of Counting in the Hierarchy

Note, that the hierarchy defined by the sub-classes $\mathcal{QCQ}^{C^k}$ by using k as a parameter, has some similar properties to the hierarchy defined by $\mathcal{QCQ}^k$ classes of ([Tur01a], [Tur04]). Both are strict and properly contained in $\mathcal{CQ}$, both are orthogonal to the Turing machine complexity hierarchy, and are characterized syntactically by machines which have a very similar behaviour. However, the expressive power of every logic C^k is much bigger than the expressive power of the corresponding logic FO^k ([Ott97]). It turns out that the sub-classes $\mathcal{QCQ}^k$ are "*very small*", while the sub-classes $\mathcal{QCQ}^{C^k}$ are "*very big*", in a certain precise sense which we define below. This fact can be clearly noted by using the notion of asymptotic probability (see Section 2). Recall from ([Tur01a], [Tur04]) that each sub-class $\mathcal{QCQ}^k$, as well as the whole hierarchy $\mathcal{QCQ}^\omega$, have a 0–1 Law. This implies a strong limitation with respect to expressive power. It is well known that a query as simple as the *parity* query (i.e., $q(I) = true$ iff $|dom(I)|$ is even) has not a 0–1 Law, and this means that this query does not even belong to the whole hierarchy $\mathcal{QCQ}^\omega$. On the other hand, the parity query belongs to the first layer of the hierarchy $\mathcal{QCQ}^{C^\omega}$. Recall the second part of the proof of Theorem 13. There, we defined a machine RCM^k which in first place computed the size of the input database. This was done using a dynamic query in C^1, so that the parity query can be computed by an RCM^1 machine. Hence, it belongs to the sub-class $\mathcal{QCQ}^{C^1}$. Then the following result follows immediately.

Proposition 32. *For any $k \geq 1$, $\mathcal{QCQ}^{C^k}$ does not have a 0–1 Law. Hence, the whole hierarchy $\mathcal{QCQ}^{C^\omega}$ does not have a 0–1 Law either.* □

Propositions 33 and 35 below (see [Gro98a]) will help us to understand the difference in the expressiveness of the corresponding sub-classes, as well as of the hierarchies. Proposition 33 is obtained as a corollary to the combination of a result by L. Babai, P. Erdős, and S. Selkow [BES80], and a result by N. Immerman and

E. Lander [IL90]. Proposition 35 can be also found in [Gro98a], and follows from results of P. Kolaitis and M. Vardi [KV92].

Proposition 33. *([BES80] and [IL90]) There is a class C of graphs with $\mu_C = 1$ such that for all graphs $I, J \in C$ we have $I \simeq J \iff I \equiv_{C^2} J$. Moreover, for all $I \in C$ and $a, b \in dom(I)$, there is an automorphism mapping a to b iff $tp_I^{C^2}(a) = tp_I^{C^2}(b)$.* □

Note that the class C of Proposition 33 is C^2-homogeneous, moreover, it is also strongly C^2-homogeneous. So, the following corollary is immediate.

Corollary 34.

(i) Almost all graphs are strongly C^2-homogeneous.

(ii) Almost all graphs which are C^2-homogeneous are also strongly C^2-homogeneous. □

Further note that this result is quite relevant not only in our context, but also in complexity theory, since the isomorphism problem is well known to be in NP, whereas M. Grohe has proved that for every $k \geq 1$, C^k equivalence is $PTIME$ complete under quantifier free reductions ([Gro96]). Examples of classes of well known graphs, though not having asymptotic probability 1, where C^k equivalence coincides with isomorphism are the class of planar graphs ([Gro98b]) and the class of trees ([IL90]).

On the other hand, the class of linear graphs is an example of a class where FO^2 equivalence coincides with isomorphism (see [EF99]).

Proposition 35. *([KV92]) Let $k \geq 1$. If C is a class of graphs such that for all graphs $I, J \in C$ we have $I \simeq J \iff I \equiv_{FO^k} J$, then $\mu_C = 0$.* □

Following [HKL96], though using a slightly different perspective, we define the notion of equality of queries *almost everywhere*. Let σ be a schema, and let q,q' be two computable queries of schema σ. Let $\mu_{(q=q')}$ be as follows:

$$\mu_{(q=q')} = \lim_{n\to\infty} \frac{|\{I \in \mathcal{B}_\sigma : dom(I) = \{1,\ldots,n\} \wedge q(I) = q'(I)\}|}{|\{I \in \mathcal{B}_\sigma : dom(I) = \{1,\ldots,n\}\}|}$$

By Proposition 33 for *every* computable query q there is a query q' in $\mathcal{QCQ}^{C^2}$ (and, hence in each layer $\mathcal{QCQ}^{C^k}$, for $k \geq 2$) such that $\mu_{(q=q')} = 1$, i.e., such that q' coincides with q over *almost all* databases. On the other hand, by Proposition 35, this cannot be true for any layer in $\mathcal{QCQ}^{\omega}$, and not even for the whole hierarchy.

On the other hand, regarding expressive power, the following result shows that the number of variables allowed in formulas is more relevant than allowing the formulas to be infinite and allowing the use of counting quantifiers. Moreover, as it is well known (see [Gro98a]), if we consider only *ordered* databases, then $\mathcal{L}^{\omega}_{\infty\omega} = C^{\omega}_{\infty\omega}$, and hence, $\mathcal{QCQ}^{\omega} = \mathcal{QCQ}^{C^{\omega}}$.

Proposition 36. *([Ott97]) For every $k \geq 1$, there is a Boolean query which is expressible in FO^{k+1}, but which is not expressible in $C^k_{\infty\omega}$.* □

As every query which is expressible in FO is clearly recursive, by using Theorem 13 and Theorem 28 we get the following corollary:

Corollary 37. *For every $k \geq 1$, $\mathcal{QCQ}^{k+1} \not\subseteq \mathcal{QCQ}^{C^k}$.* □

On the other hand, if we fix the number of variables, the strict inclusion is straightforward.

Proposition 38. *For every $k \geq 1$, $\mathcal{QCQ}^k \subset \mathcal{QCQ}^{C^k}$.*

Proof. The inclusion follows from Theorems 28 and 29. As to the strict inclusion, note that for every $k \geq 1$, an RCM^k machine can compute the C^k-type index for k-tuples of its input (see Remark 22), while clearly no RRM^k machine can do it for arbitrary databases. □

Finally, we give some examples of known classifications of queries in the infinitary logics $C^\omega_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ and, hence, by Theorems 28 and 29, in the hierarchies $\mathcal{QCQ}^{C^\omega}$ and $\mathcal{QCQ}^\omega$, respectively.

Example 39.

1): *The size of a database is even* $\in \mathcal{QCQ}^{C^1}$ and $\notin \mathcal{QCQ}^\omega$ (see observation before Proposition 32).

2): *A graph is regular* $\in \mathcal{QCQ}^{C^2}$ and $\notin \mathcal{QCQ}^\omega$ ([Ott97]).

3): *A graph is Eulerian* $\in \mathcal{QCQ}^{C^2}$ and $\notin \mathcal{QCQ}^\omega$ ([Ott97]).

4): *A graph is a disjoint union of an even number of cliques* $\in \mathcal{QCQ}^{C^2}$ and $\notin \mathcal{QCQ}^\omega$ ([KV95]).

5): *A graph is connected* $\in \mathcal{QCQ}^3$ and $\notin \mathcal{QCQ}^{C^2}$ ([Gro98a]).

6): *A graph has an even number of connected components* $\in \mathcal{QCQ}^{C^\omega}$ and $\notin \mathcal{QCQ}^\omega$ ([KV95]).

7): *A projective plane is Desargian* $\notin \mathcal{QCQ}^{C^3}$ ([Gro98a], see there also the definition of projective planes).

As we pointed out in Remark 17 (see also [Tur01a], [Tur04]), the hierarchies $\mathcal{QCQ}^\omega$ and $\mathcal{QCQ}^{C^\omega}$ are *orthogonal* to the hierarchy of complexity classes defined in terms of TIME and SPACE in Turing machine complexity. Then, we can use these hierarchies to refine the TIME and SPACE complexity classes by intersecting these classes with the different hierarchies $\mathcal{QCQ}^\omega$ and $\mathcal{QCQ}^{C^\omega}$. In this way, we obtain complexity classes which are much finer. This may result in a deeper and more

subtle understanding on the nature of queries. Next, we give some examples to illustrate this point.

Combining results from [KL99] and [KL00] with known results in descriptive complexity and with Theorem 29, we get that the following Boolean queries, defined in the class of finite groups, belong to the sub-class ($\mathcal{QCQ}^4 \cap NLOGSPACE$): *1): Given a group G, it is completely reducible and centreless; 2): Given a group G and a pair of elements a, b, the subgroup generated by a is a subgroup of the subgroup generated by b; 3): Given a group G and a pair of elements a, b, the two elements generate the same subgroup.*

As to the hierarchy $\mathcal{QCQ}^{C^\omega}$, combining known results in descriptive complexity (see [Ott97]) with results from computational complexity and with Theorem 28, we have that the following two Boolean queries belong to the sub-class ($\mathcal{QCQ}^{C^2} \cap PTIME$): *1): A graph is Eulerian; 2) A graph is regular.*

Acknowledgements

I am deeply grateful to Lauri Hella for the interesting and stimulating discussions we had on this subject. The comments of an anonymous referee were very helpful in improving both the contents and the presentation of this article.

References

[AHV94] Abiteboul, S., Hull, R. and Vianu, V., *Foundations of Databases*, Addison-Wesley, 1994.

[APV98] Abiteboul, S., Papadimitriou, C. and Vianu, V., "Reflective Relational Machines", *Information and Computation* 143, pp. 110-136, 1998.

[AV95] Abiteboul, S., Vianu, V., "Computing with first-order logic", *Journal of Computer and System Sciences* 50(2), pp. 309-335, 1995.

[AVV95] Abiteboul, S., Vardi, M. and Vianu, V., "Computing with Infinitary Logic", *Theoretical Computer Science* 149(1), pp. 101-128, 1995.

[AVV97] Abiteboul, S., Vardi, M. and Vianu, V., "Fixpoint Logics, Relational Machines, and Computational Complexity", *Journal of ACM* 44(1), pp. 30-56, 1997.

[BES80] Babai, L., Erdös, P. and Selkow, S., "Random Graph Isomorphism". *SIAM Journal on Computing* 9, pp. 628-635, 1980.

[CFI92] Cai, J. Y., Fürer, M. and Immerman, N., "An Optimal Lower Bound on the Number of Variables for Graph Identification". *Combinatorica* 12(4), pp. 389-410, 1992.

[CH80] Chandra, A. K., Harel, D., "Computable Queries for Relational Data Bases", *Journal of Computer and System Sciences* 21(2), pp. 156-178, 1980.

[CK92] Chang, C. and Keisler, H., *Model Theory*, 3rd ed., Elsevier North Holland, 1992.

[DLW95] Dawar, A., Lindell, S., Weinstein, S., "Infinitary Logic and Inductive Definability over Finite Structures", *Information and Computation* 119(2), pp. 160-175, 1995.

[Ebb85] Ebbinghaus, H., "Extended Logics: The General Framework", in *Model Theoretic Logics*, ed. Barwise and Fefferman, Springer-Verlag, pp. 25-76, 1985.

[EF99] Ebbinghaus, H., Flum, J., *Finite Model Theory*, Springer, 2nd. ed., 1999.

[GM98] Grohe, M. and Mariño, J., "Definability and Descriptive Complexity on Databases of Bounded Tree-Width", in Proceedings of International Conference on Database Theory, ICDT 1999, Springer, LNCS 1540, pp. 70-82, 1998.

[Gro96] Grohe, M., "Equivalence in Finite Variable Logics is Complete for Polynomial Time", in Proceedings of 37th IEEE Symposium on Foundations of Computer Science, pp. 264-273, 1996.

[Gro98a] Grohe, M., "Finite Variable Logics in Descriptive Complexity Theory", Preliminary version, 1998.

[Gro98b] Grohe, M., "Fixed Point Logics on Planar Graphs", in Proceedings of 13th IEEE Symposium on Logic in Computer Science, LICS 1998, pp. 6-15, 1998.

[Hel96] Hella, L., "Logical Hierarchies in PTIME", *Information and Computation* 129(1), pp. 1-19, 1996.

[HKL96] Hella, L., Kolaitis, P. and Luosto, K., "Almost Everywhere Equivalence of Logics in Finite Model Theory", *The Bulletin of Symbolic Logic* 2(4), pp. 422-443, 1996.

[IL90] Immerman, N. and Lander, E., "Describing Graphs: A First Order Approach to Graph Canonization", in *Complexity Theory Retrospective*, ed. A. Selman, Springer, pp. 59-81, 1990.

[Imm99] Immerman, N., *Descriptive Complexity*, Springer, 1999.

[KL99] Koponen, A. and Luosto, K., "Definability of Group Theoretic notions", *Research Report of the Department of Mathematics of the University of Helsinki* 227, 1999.

[KL00] Koponen, A. and Luosto, K., personal comunication, 2000.

[KV92] Kolaitis, P. and Vardi, M., "Infinitary Logic and 0–1 Laws", *Information and Computation* 98, pp. 258-294, 1992.

[KV95] Kolaitis, P. and Väänänen, J., "Generalized Quantifiers and Pebble Games on Finite Structures", *Annals of Pure and Applied Logic* 74, pp. 23-75, 1995.

[Ott96] Otto, M., "The Expressive Power of Fixed Point Logic with Counting", *Journal of Symbolic Logic* 61 (1), pp. 147-176, 1996.

[Ott97] Otto, M., *Bounded Variable Logics and Counting*, Springer, 1997.

[Tur96] Turull Torres, J.M., "Partial Distinguishability and Query Computability on Finite Structures", communicated in the XI *Brazilean Symposium on Logic*, XI-EBL, Salvador, Brazil, 1996. Abstract published in the Bulletin of the IGPL, London, 3-1996.

[Tur98] Turull Torres, J.M., "Query Completeness, Distinguishability and Relational Machines", in *Models, Algebras and Proofs*: Selected Papers from the X Latin American Symposium on Mathematical Logic, Bogotá 1995, ed. Marcel-Dekker, pp. 135-163, 1998.

[Tur01a] Turull Torres, J. M., "A Study of Homogeneity in Relational Databases", *Annals of Mathematics and Artificial Intelligence* 33(2), pp. 379-414, 2001.

[Tur01b] Turull Torres, J. M., "Semantic Classifications of Queries to Relational Databases", in *Semantics in Databases*, L. Bertossi, G. Katona, K.-D. Schewe, B. Thalheim (Eds.), Springer LNCS.

[Tur02] Turull Torres, J. M., "Relational Databases and Homogeneity in Logics with Counting", in Foundations of Information and Knowledge Systems: Second International Symposium, FoIKS 2002, Proceedings, Germany, Springer LNCS 2284, 2002.

[Tur04] Turull Torres, J. M., "Erratum for: A Study of Homogeneity in Relational Databases [Annals of Mathematics and Artificial Intelligence 33(2) (2001) 379-414]", *Annals of Mathematics and Artificial Intelligence* 42(4), pp. 443-444, 2004.

Received August, 2004

Acta Cybernetica **17** (2006) 513–519.

On a Class of Discrete Functions

Dimiter Stoichkov Kovachev*

Abstract

We consider classes of functions which depend in a certain way on their variables. The relation between the number of *H-functions* of n variables of the *k-valued logic* and the number of *n-dimensional Latin hypercubes* of order k is found. We have shown how from an arbitrary *Latin hypercube* we can "construct" (present in table form) an *H-function* and vice versa - how every *H-function* can be represented as a *Latin hypercube*. We extend the concepts of *H-function* and *Latin hypercube*.

Keywords: H-function, subfunction, range, spectrum, Latin hypercube.

1 Introduction

In the paper we interpret the *H-function* as Latin squares or Latin hypercubes. On the other side the Latin squares and Latin hypercubes are well known combinatorial structures which are widely used in different areas of mathematics and its applications, in theoretical and applied computer science, etc. They are very important in Statistics, Coding Theory, Cryptography, Tournament Design, etc. ([3], see §1.4, §12.1 - 12.4; §1.5, §13.1 - 13.5; §14.1 - 14.4; §1.6, §16.5, respectively), Design Experiment, Security of Information, Decision Making, etc.

Let $P_n^k = \{f : E_k^n \longrightarrow E_k / E_k = \{0, 1, ..., k-1\}, k \geq 2\}$.

Definition 1. *[1] We say that* $f(x_1, x_2, ..., x_n)$ *is* H-function *if for every variable* x_i, $1 \leq i \leq n$, $n \geq 2$ *and for every* $a_1, ..., a_{i-1}, a', a'', a_{i+1}, ..., a_n \in E_k$ *for* $a' \neq a''$ *we have* $f(a_1, ..., a_{i-1}, a', a_{i+1}, ..., a_n) \neq f(a_1, ..., a_{i-1}, a'', a_{i+1}, ..., a_n)$.

Definition 2. *[4] The number* $Rng(f)$ *of different values of the function* f *is called* range *of* f.

Denote by X_f and $P_n^{k,q}$ respectively, the set of variables of function f, and the set of all functions of P_n^k with range q, $1 \leq q \leq k$.

Definition 3. *[2] The function* h *is called a* subfunction *of the function* $f(x_1, x_2, ..., x_n)$ *with respect to the set of variables* $R = \{x_{j_1}, x_{j_2}, ..., x_{j_r}\}$,

*Department of computer science, South-West University "N. Rilski", 2700 Blagoevgrad, P.O.79, E-mail: dkovach@aix.swu.bg and dkovach@abv.bg

$R \subseteq X_f$, if h is obtained from f by replacement the variables from R respectively by values $c_1, c_2, \ldots, c_r$. We will denote the subfunction h in one of the following ways $h \overset{R}{\prec} f$ or $h = f(x_{j_1} = c_1, x_{j_2} = c_2, \ldots, x_{j_r} = c_r)$.

Let M, $M \subseteq X_f$, be a set of variables and G be the set of all subfunctions of f with respect to $X_f \backslash M$, i.e. $G = G(M, f) = \{g : g \overset{X_f \backslash M}{\prec} f\}$.

Definition 4. *[4] The set $Spr(M, f) = \bigcup_{g \in G}\{Rng(g)\}$ is called* spectrum *of the set M for the function f.*

A matrix B with m rows and m columns is denoted by $B = (b_{ij})_1^m$. Matrix $A = (a_{j_1 j_2 \ldots j_n})_1^k$ is an n-dimensional matrix of order k.

Definition 5. *Latin n-dimensional hypercube of order k based on the set E_k is defined as every matrix $A = (a_{j_1 j_2 \ldots j_n})_1^k$, such that for every s , s=1, 2,..., n we have*

$$\{a_{i_1 \ldots i_{s-1}\ 1\ i_{s+1} \ldots i_n}\} \bigcup \{a_{i_1 \ldots i_{s-1}\ 2\ i_{s+1} \ldots i_n}\} \bigcup \ldots \bigcup \{a_{i_1 \ldots i_{s-1}\ \mathbf{k}\ i_{s+1} \ldots i_n}\} = E_k,$$

i.e.

$$\Big|\bigcup_{j=1}^{k} \{a_{i_1 \ldots i_{s-1}\ \mathbf{j}\ i_{s+1} \ldots i_n}\}\Big| = |E_k| = k. \tag{1}$$

The set of all Latin n-dimensional hypercubes of order k will be denoted by LHC_n^k.

Theorem 6. *Each matrix $A = (a_{j_1 j_2 \ldots j_n})_1^k$, for which*

$$a_{j_1 j_2 \ldots j_n} = [\sum_{r=1}^{n} f_r(j_r - 1) + c\,]\ mod\ k,$$

where $f_r \in P_1^{k,k}$, c is a natural number, belongs to LHC_n^k.

Proof. Each function $h \in P_1^{k,k}$, is of the form $h = \begin{pmatrix} 0 & 1 & \ldots & k-1 \\ l_1 & l_2 & \ldots & l_k \end{pmatrix}$, where $l_1, l_2, \ldots, l_k$ is a permutation of the numbers $0, 1, \ldots, k-1$, and $h(t) = l_{t+1}$, $t = 0, 1, \ldots, k-1$. Assume that the matrix $A \notin LHC_n^k$ and there exists s, $s \in \{1, 2, \ldots, n\}$, such that $|\bigcup_{j=1}^{k}\{a_{i_1 \ldots i_{s-1}\ \mathbf{j}\ i_{s+1} \ldots i_n}\}| \neq E_k$. Therefore there exist α, β, $\alpha \neq \beta$, such that $a_{i_1 \ldots i_{s-1}\ \alpha\ i_{s+1} \ldots i_n} = a_{i_1 \ldots i_{s-1}\ \beta\ i_{s+1} \ldots i_n}$. From the last equality it follows that $f_s(\alpha - 1) = f_s(\beta - 1)$ and from $f_s \in P_1^{k,k}$, we have $\alpha = \beta$: a contradiction. Therefore $A \in LHC_n^k$. □

Function $h_1(x) = (ax + b)\ mod\ k$, where a, b are natural numbers, $(a, k) = 1$ belongs to $P_1^{k,k}$. As a corollary of Theorem 6 we obtain that matrix $B = (b_{j_1 j_2 \ldots j_n})_1^k$, for which $b_{j_1 j_2 \ldots j_n} = (a_1 j_1 + a_2 j_2 + \ldots + a_n j_n + c)\ mod\ k$, for $(a_i, k) = 1$, $i = 1, 2, \ldots, n$ belongs to LHC_n^k.

Each function from $P_1^{k,k}$ can be represented by a table or by interpolating polynomial and, based on Theorem 6, can be used for "constructing" elements of LHC_n^k.

Example 7. Construct Latin 2-dimensional hypercube of order 4 and Latin 3-dimensional hypercube of order 3.

Let $f_1 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 0 & 2 & 1 \end{pmatrix}$ and $f_2 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 1 & 0 \end{pmatrix}$, be arbitrary functions from $P_1^{4,4}$. Let matrix $C = (c_{ij})_1^4$ be such that $c_{ij} = [f_1(i-1) + f_2(j-1)]$ *mod* k. Then $c_{11} = [f_1(1-1) + f_2(1-1)]$ *mod* $4 = [3+2]$ *mod* $4 = 1$. Similarly we obtain the remaining elements of matrix C. So we have

$$C = (c_{ij})_1^4 = \begin{pmatrix} 1 & 2 & 0 & 3 \\ 2 & 3 & 1 & 0 \\ 0 & 1 & 3 & 2 \\ 3 & 0 & 2 & 1 \end{pmatrix} \in LHC_2^4;\ D = (d_{ijl})_1^3,\ D \in LHC_3^3,$$

where $d_{ijl} = (2i + j + 2l + 1)$ *mod* 3.

2 Spectrum of H-functions and Latin Hypercubes

Theorem 8. *The function* $f(x_1, x_2, ..., x_n) \in P_n^k$ *is H-function if and only if for each variable* x_i, $i = 1, 2, ..., n$ *we have* $Spr(x_i, f) = \{k\}$.

Proof. (Necessity) Let f be H-function, x_i be arbitrary variable of f. Then for every set of constants $a_1, ..., a_{i-1}, a_{i+1}, ..., a_n$ we have

$$f(a_1, ..., a_{i-1}, r, a_{i+1}, ..., a_n) \neq f(a_1, ..., a_{i-1}, t, a_{i+1}, ..., a_n) \tag{2}$$

for each r and t for which $r \neq t$, $r, t \in E_k$.

From Definition 3 and the inequality (2) it follows that every subfunction of f with respect to $X_f \backslash \{x_i\}$ assumes exactly k different values, i.e. it has a range equal to k.

For $M = \{x_i\}$, from Definition 4 it follows that $Spr(x_i, f) = \{k\}$.

(Sufficiency) Let for the variable x_i we have

$$Spr(x_i, f) = \{k\}. \tag{3}$$

From Definition 4 for $M = \{x_i\}$ and (3) it follows that every subfunction of f with respect to $X_f \backslash \{x_i\}$ has a range equal to k. This means that for an arbitrary $n-1$ tuple of values $< c_1, ..., c_{i-1}, c_{i+1}, ..., c_n >$, the subfunction $f(x_1 = c_1, ..., x_{i-1} = c_{i-1}, x_i, x_{i+1} = c_{i+1}, ..., x_n = c_n)$ has a range k, i.e. it assumes exactly k different values. Because x_i can also assume exactly k different values, we obtain that for every $c', c'' \in E_k$, $(c' \neq c'')$, the following inequality holds $f(c_1, ..., c_{i-1}, c', c_{i+1}, ..., c_n) \neq f(c_1, ..., c_{i-1}, c'', c_{i+1}, ..., c_n)$.

Since the variable x_i and the set of values $< c_1, ..., c_{i-1}, c', c'', c_{i+1}, ..., c_n >$ are arbitrary, from Definition 1 it follows that f is an H-function. □

Remark 9. More that Theorem 8 can also be used as a definition of an H-function, where the restriction $n \geq 2$ can be eliminated, i.e. the definition holds for the functions of one variable as well.

Theorem 10. *The number of all H-functions of P_n^k is equal to the number of all Latin n-dimensional hypercubes of order k.*

Proof. Let $< c_{1i},\ c_{2i},...,\ c_{ni} >$, i=1, 2,..., k^n be all the possible $n-$tuples of constants and $f \in P_n^k$ be an arbitrary function for which

$$f(x_1 = c_{1i},\ x_2 = c_{2i},...,\ x_n = c_{ni}) = a_i,\ i = 1,\ 2,...,\ k^n,\ a_i \in E_k. \qquad (4)$$

Let the mapping $\varphi_f : E_k^{n+1} \longrightarrow E_k$ be such that it maps a_i from any equality (4) into an element of the matrix $D_1 = (d_{j_1 j_2 ... j_n})_1^k$,

$$d_{j_{1i} j_{2i} ... j_{ni}} = a_i = \varphi_f(c_{1i},\ c_{2i},...,\ c_{ni},\ a_i),\ i = 1,\ 2,...,\ k^n, \qquad (5)$$

where

$$j_{1i} = c_{1i} + 1,\ j_{2i} = c_{2i} + 1,...,\ j_{ni} = c_{ni} + 1,\ i = 1,\ 2,...,\ k^n. \qquad (6)$$

Conversely, to every element of the matrix D_1 from (5), through the equalities (6), the constant a_i from equality (4) is assigned uniquely.

If we take into consideration Definitions 1 and 5 as well, we draw the conclusion that to each H-function of P_n^k we can assign, using φ_f, a Latin n-dimensional hypercube of order k, and vice versa. Therefore, the number of H-functions of P_n^k is equal to the number of the Latin n-dimensional hypercubes of order k. □

Using the mapping φ_f, every Latin n-dimensional hypercube of order k, which elements are in the set E_k, can be used for the "construction"(i.e. tabular representation) of an H-function of P_n^k.

Corollary 11. *Every H-function of P_n^k can be represented as Latin n-dimensional hypercube of order k and vice versa, every Latin $n-$dimensional hypercube of order k can be represented as H-function.*

In the general case, the sum of the H-functions of P_n^k can be an H-function, but this is not always true.

Example 12. Indeed, if $f_1 \in P_n^k$ is an arbitrary H-function, then $f_2 = k - 1 - f_1$ is also an H-function. However the sum $f_1 + f_2 = k - 1$ is a constant and it is not an H-function.

Example 13. Let k be an odd number and $f \in P_n^k$ be an arbitrary H-function. For every number $a \in E_k = \{0,\ 1,...,\ k-1\}$ the function f_a, $f_a = f + a$ (mod k) is also an H-function. In addition, the sum $f + f_a = 2.f + a$ (mod k) is an H-function.

The problem for finding necessary and sufficient conditions under which the sum of two H-functions is again an H-function remains open.

From Definitions 1, 2, 3 it follows that a function of P_n^k is an H-function if each of its subfunctions of one variable takes k different values, i.e. if it has a range equal to k.

3 Generalizations of H-functions

We extend the concept of H-function in two directions - increasing the number of the variables on which the function depends in a certain way (each of its subfunctions of $m \geq 2$ variables takes q, $1 \leq q \leq k$, different values, i.e. it has a range equal to q) and changing the number of different values which the function assumes in this dependence.

Let m, q be integers such that $1 \leq m \leq n$, $1 \leq q \leq k$, and M be an arbitrary set of m variables of the function $f \in P_n^k$.

Definition 14. *We say that the function $f(x_1, x_2, ..., x_n) \in P_n^k$ is an $H[m;q]$-function, if for every set M, $M \subseteq X_f$, $|M| = m$, we have*

$$Spr(M, f) = \{q\}. \tag{7}$$

The set of all functions of P_n^k for which (7) holds will be denoted by $H[m;q]_n^k$. When $m = k$, the set $H[1;m]_n^k$ coincides with the set of all H-functions from P_n^k.

From Definition 14 it follows that a function of P_n^k is an $H[m;q]$ function if each of its subfunctions of m variables takes q different values, i.e. it has a range equal to q.

We will prove a necessary and sufficient condition for a function to be an $H[m;q]$ function, which generalizes Theorem 8.

Theorem 15. *A function $f \in P_n^k$ is an $H[m;q]$ function if and only if each of its subfunctions, depending on at least m variables, is an $H[m;q]$ function.*

Proof. (Necessity) Let $f \in P_n^k$ be an $H[m;q]$ function and let h be an arbitrary subfunction of f, for which $|X_h| \geq m$. We will prove that h is an $H[m;q]$ function. Let us suppose that h is not an $H[m;q]$ function. Therefore there is a set of variables M, $|M| = m$, $M \subseteq X_h$, such that

$$Spr(M, h) \neq \{q\}. \tag{8}$$

From (8) it follows that a subfunction h_1 exists, $h_1 \overset{X_h \setminus M}{\prec} h$, such that $Rng(h_1) \neq q$. Since $h_1 \overset{X_h \setminus M}{\prec} h$, $h \prec f$, it follows that $h_1 \overset{X_f \setminus M}{\prec} f$ and $Rng(h_1) \neq q$.
From Definition 4 and Definition 14 it follows that $Spr(M, f) \neq \{q\}$ and f is not an $H[m;q]$ function. This is a contradiction.
(Sufficiency) Let each subfunction of f, depending on at least m variables, be an $H[m;q]$-function. We will prove that f is an $H[m;q]$-function. Let us suppose that f is not an $H[m;q]$-function, i.e. there exists a set of variables M, $|M| = m$, $M \subseteq X_f$, such that

$$Spr(M, f) \neq \{q\}. \tag{9}$$

From (9) it follows that there is a subfunction g, $g \overset{X_f \setminus M}{\prec} f$, $|X_g| = m$, for which $Rng(g) \neq q$. Therefore the subfunction g is not an $H[m;q]$-function which contradicts the given condition. The contradiction is due to the assumption that f is not an $H[m;q]$-function. □

As a corollary of Theorem 15 for $m = 1$, $q = k$ we get:

Corollary 16. *A necessary and sufficient condition for the function $f \in P_n^k$ to be an H-function is that every of its subfunctions, depending on at least one variable, is an H-function.*

Definition 17. *An n-dimensional matrix $W = (w_{i_1 i_2 \ldots i_n})_1^k$ of order k, the elements of which are in the set E_k, such that when we fix arbitrary $n - m$ of its indices, we get an m-dimensional matrix of order k, in which there are exactly q different elements of the set E_k, is called an n-dimensional $H[m;q]$-hypercube of order k, generated by E_k.*

The set of all n-dimensional $H[m;q]$-hypercubes of order k, generated by the set E_k, will be denoted by $HHC[m;q]_n^k$.

It is obvious that for m=1, $q = k$, $LHC_n^k = HHC[1;k]_n^k$ holds, i.e. the Latin n-dimensional hypercubes of order k are special cases of the n-dimensional $H[m;q]$-hypercubes of order k.

Example 18. If the elements of the matrix of an arbitrary Latin hypercube of the set LHC_n^k are taken by modulo q, then we will get the matrix of an $H[1;q]$-hypercube of the set $HHC[1;q]_n^k$. In the matrix obtained in this way there will be exactly q different elements of E_k.

Let the matrix C from Example 7 be taken by modulo 3 and the new matrix we get be denoted by C_1

$$C = \begin{pmatrix} 1 & 2 & 0 & 3 \\ 2 & 3 & 1 & 0 \\ 0 & 1 & 3 & 2 \\ 3 & 0 & 2 & 1 \end{pmatrix} \in LHC_2^4,\ C \xrightarrow[mod\ 3]{q=3} C_1 = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 2 & 1 \end{pmatrix} \in HHC[1;3]_2^4.$$

Of course, there are matrices in the set $HHC[1;q]_n^k$ which cannot be obtained from matrices of the set LHC_n^k by taking modulo q. Below in Example 20 we have shown such a matrix. The matrix B is constructed in which the number of different elements is more that q, $(q = 3)$.

Theorem 19. *Each matrix $B = (b_{j_1 j_2 \ldots j_n})_1^k$, for which*

$$b_{j_1 j_2 \ldots j_n} = [\sum_{r=1}^{n} g_r(j_r - 1)]\ mod\ k,$$

where $g_r \in P_1^{k,q}$, $r = 1, 2, \ldots, n$, is an n-dimensional $H[1;q]$-hypercube of order k.

Proof. If $f_1 \in P_1^{k,q}$ then f_2, ($f_2 = (f_1 + c_0)\ mod\ k$, c_0 is a natural number) also belongs to $P_1^{k,q}$. By fixing any $n - 1$ indices of the matrix B we will obtain a function of $P_1^{k,q}$ and according to Definition 17, $B \in HHC[1;q]_n^k$. □

Because each function $g_r \in P_1^{k,q}$, $r = 1, 2, \ldots, n$, can be chosen in $\mid P_1^{k,q} \mid$ ways it follows that $\mid HHC[1;q]_n^k \mid \leq \mid P_1^{k,q} \mid^n$.

Example 20. Construct 2-dimensional $H[1;3]$ -hypercube of order 4. Let $g_1 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 2 & 1 \end{pmatrix}$ and $g_2 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 \end{pmatrix}$, be arbitrary functions from $P_1^{4,3}$ Let matrix $B = (b_{ij})_1^4$ be such that $b_{ij} = [g_1(i-1) + g_2(j-1)]$ *mod* 4. We compute the elements of the matrix B and we get:

$$B = (b_{ij})_1^4 = \begin{pmatrix} 1 & 0 & 3 & 0 \\ 0 & 3 & 2 & 3 \\ 0 & 3 & 2 & 3 \\ 3 & 2 & 1 & 2 \end{pmatrix} \in HHC[1;3]_2^4,\ m = 1,\ n = 2,\ q = 3,\ k = 4.$$

Proposition 21. *The number of hypercubes of the set $HHC[m;q]_n^k$ is equal to the number of functions of the set $H[m;q]_n^k$, i.e.*

$$|HHC[m;q]_n^k| = |H[m;q]_n^k|.$$

Using Definitions 2, 3, 4, 14, 17 and the arguments from Theorem 8 we can complete the proof of Proposition 21.

Every n-dimensional $H[m;q]$-hypercube of order k generated by the set of k elements E_k can be used for the "construction"(i.e. tabular representation) of a function of the set $H[m;q]_n^k$.

References

[1] Chimev K. N. *On a way some functions of P_k depend on their arguments.* , Annuaire Des Ecoles Techniques Superieures, Mathematique, vol. IV, livre. 1, pp. 5-12, 1967.

[2] Chimev K. N. *Discrete Functions and Subfunctions*, Blagoevgrad, 1991, (in Bulgarian).

[3] Laywine Ch. F., Mullen G. L. *Discrete Mathematics Using Latin Squares*, John Wiley & Sons, New York, 1998.

[4] Kovachev D. S. *On the Number of Some k-Valued Functions of n Variables*, Union of Bulgarian Mathematicians, Mathematics and Education in Mathematics, Proceedings of Thirtieth Spring Conference of the Union of Bulgarian Mathematicians, Borovets, pp. 176-181, April 8-11, 2001.

[5] Kovachev D. S. *On the Number of Discrete Functions with a Given Range*, General Algebra and Applications, Proceedings of "59th Workshop on General Algebra", Potsdam edited by K. Denecke and H.-J. Vogel, pp.125-134, 2000.

Received June, 2005

Acta Cybernetica **17** (2006) 521–531.

On Armstrong Relations for Strong Dependencies

Vu Duc Thi* and Nguyen Hoang Son†

Abstract

The strong dependency has been introduced and axiomatized in [2], [3], [4], [5]. The aim of this paper is to investigate on Armstrong relations for strong dependencies. We give a necessary and sufficient condition for an abitrary relation to be Armstrong relation of a given strong scheme. We also give an effective algorithm finding a relation r such that r is Armstrong relation of a given strong scheme $G = (U, S)$ (i.e. $S_r = S^+$, where S_r is a full family of strong dependencies of r, and S^+ is a set of all strong dependencies that can be derived from S by the system of axioms). We estimate this algorithm. We show that the time complexity of this algorithm is polynomial in $|U|$ and $|S|$.

1 Introduction

Let us give some necessary definitions and results that are used in next section.

Definition 1. *Let U be a nonempty finite set of attributes, $r = \{h_1, \ldots, h_m\}$ a relation over U, and $A, B \subseteq U$. We say that B strongly depends on A in r (denote $A \xrightarrow[r]{s} B$) iff*

$$(\forall h_i, h_j \in r)((\exists a \in A)(h_i(a) = h_j(a) \Rightarrow (\forall b \in B)(h_i(b) = h_j(b))).$$

Let $S_r = \{(A, B) : A \xrightarrow[r]{s} B\}$. S_r is called a full family of strong dependencies of r. Where we write (A, B) or $A \to B$ for $A \xrightarrow[r]{s} B$ when r, s are clear from the context.

Definition 2. *A strong dependency (SD) over U is a statement of form $X \to Y$, where $X, Y \subseteq U$. The SD $X \to Y$ holds in a relation r if $A \xrightarrow[r]{s} B$. We also say that r satisfies the SD $A \to B$.*

Definition 3. *Let U be a set of attributes and $\mathcal{P}(U)$ its power set. Let $Y \subseteq \mathcal{P}(U) \times \mathcal{P}(U)$. We say that Y is an s - family over U iff for all $A, B, C, D \subseteq U$ and $a \in U$*

*Institute of Information Technology, Vietnamese Academy of Science and Technology, 18 Hoang Quoc Viet, Hanoi, Vietnam

†Department of Mathematics, College of Sciences, Hue University, Vietnam

(S1) $(\{a\},\{a\}) \in Y$,
(S2) $(A,B) \in Y, (B,C) \in Y, B \neq \emptyset \Rightarrow (A,C) \in Y$,
(S3) $(A,B) \in Y, C \subseteq A, D \subseteq B \Rightarrow (C,D) \in Y$,
(S4) $(A,B) \in Y, (C,D) \in Y \Rightarrow (A \cup C, B \cap D) \in Y$,
(S5) $(A,B) \in Y, (C,D) \in Y \Rightarrow (A \cap C, B \cup D) \in Y$.

It is easy to see that S_r is an s – family over U.

It is known [4] that if Y is an s – family over U, then there exists a relation r such that $Y = S_r$.

Definition 4. *A strong scheme G is a pair (U,S), where U is a finite set of attributes, and S a set of SDs over U.*

Let S^+ be a set of all SDs that can be derived from S by the rules in Definition 3.

It can be seen [4] that if $G = (U,S)$ is a strong scheme then there is a relation r over U such that $S_r = S^+$. Such a relation is called Armstrong relation of G.

Definition 5. *Let K be a Sperner-system over U. We define the set of antikeys of K, denoted by K^{-1}, as follows:*

$$K^{-1} = \{A \subset U : (B \in K) \Rightarrow (B \not\subseteq A) \text{ and } (A \subset C) \Rightarrow (\exists B \in K)(B \subseteq C)\}.$$

Definition 6. *The mapping $F : \mathcal{P}(U) \longrightarrow \mathcal{P}(U)$ is called a strong operation over U if for every $a,b \in U$ and $A \in \mathcal{P}(U)$ the following properties hold:*

(1) $a \in F(\{a\})$,

(2) $b \in F(\{a\})$ *implies* $F(\{b\}) \subseteq F(\{a\})$,

(3) $F(A) = \bigcap_{a \in A} F(\{a\})$.

Remark 7. It is clear that for arbitrary strong operation F

(1) $F(\emptyset) = U$,

(2) For all $A,B \in \mathcal{P}(U) : F(A \cup B) = F(A) \cap F(B)$,

(3) If $A \subseteq B$ then $F(B) \subseteq F(A)$.

It can be seen that the set $\{F(\{a\}) : a \in U\}$ determines the set $\{F(A) : A \in \mathcal{P}(U)\}$.

The following theorem shows that between s – families and strong operations there exists a one - to - one correspondence

Theorem 8. [7] *Let S be a s – family over U. We define the mapping F_S as follows: $F_S(A) = \{a \in U : (A,\{a\}) \in S\}$. Then F_S is a strong operation over U. Conversely, if F is a strong operation over U then there is exactly one s – family S over U such that $F_S = F$, where $S = \{(A,B) : B \subseteq F(A)\}$.*

Definition 9. *Let $G = (U, S)$ be a strong scheme over U, $A \subseteq U$. We set*

$$A^+ = \{a \in U : A \to \{a\} \in S^+\}.$$

A^+ is called the closure of A over G.

It is clear that $A \to B \in S^+$ iff $B \subseteq A^+$.

Lemma 10. *Let $G = (U, S)$ be a strong scheme over U. Suppose that $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_l\}$ are subsets of U. Then $A \to B \in S^+$ if and only if $\{a_i\} \to \{b_j\} \in S^+$ for every $i = 1, \ldots, k; j = 1, \ldots, l$.*

Proof. By rules (S3), (S4) and (S5), the lemma is obvious. □

Algorithm 11. [6] (Finding $\{a\}^+$)

Input: given a strong scheme $G = (U, S)$, where $S = \{A_i \to B_i : i = 1, \ldots, m\}, a \in U$.

Output: compute $\{a\}^+$.

Method: we compute $\{a\}^+$ by induction.

Step 1. We set $X^{(0)} = \{a\}$.

Step i+1. If there is a SD $A_j \to B_j \in S$ so that $A_j \cap X^{(i)} \neq \emptyset$ and $B_j \not\subseteq X^{(i)}$ then we set

$$X^{(i+1)} = X^{(i)} \cup (\bigcup_{A_j \cap X^{(i)} \neq \emptyset} B_j).$$

In the converse case we set $\{a\}^+ = X^{(i)}$.

It is easy to see that there is a k such that $\{a\} = X^{(0)} \subseteq X^{(1)} \subseteq \cdots \subseteq X^{(k)} = X^{(k+1)} = \cdots$ and we set

$$\{a\}^+ = X^{(k)}.$$

Proposition 12. [6] *For each $a \in U$ Algorithm 11 computes $\{a\}^+$.*

It can be seen that the complexity of Algorithm 11 is polynomial time in the $|U|, |S|$.

Proposition 13. [6] *Let $G = (U, S)$ be a strong scheme over U, and $A \to B$ is a SD. Then there is a polynomial time algorithm deciding whether $A \to B \in S^+$.*

2 Armstrong Relation for Strong Dependency

It is known [8] that there is an algorithm that finds a set of all antikeys from a given Sperner-system.

Algorithm 14. [8]

Input: a Sperner-system $K = \{B_1, \ldots, B_m\}$ over U.

Output: K^{-1}.

Method:

Step 1. We set $K_1 = \{U - \{a\} : a \in B_1\}$. It is clear that $K_1 = \{B_1\}^{-1}$.

Step q+1 $(q < m)$. We suppose that $K_q = F_q \cup \{X_1, \ldots, X_{t_q}\}$, where $X_1, \ldots, X_{t_q}$ containing B_{q+1} and $F_q = \{A : A \in K_q, B_{q+1} \not\subseteq A\}$. For all i $(i = 1, \ldots, t_q)$ we construct the antikeys of $\{B_{q+1}\}$ on X_i in an analogous way as K_1. Denote them by $A_1^i, \ldots, A_{r_i}^i$ $(i = 1, \ldots, t_q)$. Let

$$K_{q+1} = F_q \cup \{A_p^i : A \in F_q \Rightarrow A_p^i \not\subset A, 1 \leq i \leq t_q, 1 \leq p \leq r_i\}.$$

We set $K^{-1} = K_m$.

Theorem 15. [8] *For each* q $(1 \leq q \leq m)$, $K_q = \{B_1, \ldots, B_q\}^{-1}$, *i.e.* $K_m = K^{-1}$.

It can be seen that K and K^{-1} are uniquely determined by one another and the determination of K^{-1} based on our algorithm does not depend on the order of $B_1, \ldots, B_m$. Denote $K_q = F_q \cup \{X_1, \ldots, X_{t_q}\}$ and let l_q $(1 \leq q \leq m-1)$ be the number of elements of K_q.

Proposition 16. [8] *The worst-case time complexity of our Algorithm 14 is*

$$\mathcal{O}(|U|^2 \sum_{q=1}^{m-1} t_q u_q),$$

where

$$u_q = \begin{cases} l_q - t_q & \text{if } l_q > t_q, \\ 1 & \text{if } l_q = t_q. \end{cases}$$

Note that $l_q \geq t_q$. Clearly, in each step of our algorithm K_q is a Sperner-system. In the cases for which $l_q \leq l_m (q = 1, ..., m-1)$, it is easy to see that the time complexity of our algorithm is not greater than $\mathcal{O}(|U|^2|K||K^{-1}|^2)$. Hence, in these cases Algorithm 14 finds K^{-1} in polynomial time in $|U|, |K|$ and $|K^{-1}|$. Obviously, if the number of elements of K is small, then Algorithm 14 is very effective. It only requires polynomial time in $|U|$.

Definition 17. *Let* $G = (U, S)$ *be a strong scheme over* U, *and* $a \in U$. *We set*

$$K_a = \{A \subseteq U : A \to \{a\} \in S^+, \nexists B : (B \to \{a\} \in S^+)(B \subset A)\}.$$

K_a *is called the family of minimal sets of the attribute* a.

Clearly, $\{a\} \in K_a, U \notin K_a$ and K_a is a Sperner-system over U.

Proposition 18. *Let* $G = (U, S)$ *be a strong scheme over* $U, a \in U$, K_a *is a family of minimal sets of* a *and* $n = |U|$. *Then*

(1) $K_a = \{\{b\} : b \in U, \{b\} \to \{a\} \in S^+\}$.

(2) $\forall A \in K_a : |A| = 1$.

(3) $|K_a| \leq n$.

(4) $|K_a^{-1}| = 1$.

Proof. (1) We define the mapping $F_S : \mathcal{P}(U) \longrightarrow \mathcal{P}(U)$ as follows:

$$F_S(A) = \{a \in U : A \rightarrow \{a\} \in S^+\}.$$

By Theorem 8, it is clear that F_S is a strong operation over U. It is easy to see that $A^+ = F_S(A)$. Consequently, by Definition 6 we have

$$\begin{aligned} A^+ &= \bigcap_{a \in A} F_S(\{a\}) \\ &= \bigcap_{a \in A} \{b \in U : \{a\} \rightarrow \{b\} \in S^+\} \\ &= \bigcap_{a \in A} \{a\}^+. \end{aligned} \tag{1}$$

By (1) we obtain $A^+ \subseteq \{a\}^+ \quad \forall a \in A$. From this and the definition of K_a we immediately get

$$K_a = \{\{b\} : b \in U, \{b\} \rightarrow \{a\} \in S^+\}.$$

(2) It is obvious from (1).
(3) Because for each $A \in K_a : |A| = 1$, we can be seen that $|K_a| \leq n$.
(4) By (2) and the definition of antikeys set, it is clear that $|K_a^{-1}| = 1$.
The proposition is proved. □

From this proposition we construct an algorithm finding a minimal set of the attribute a.

Algorithm 19. MSA

Input: a strong scheme $G = (U, S)$, and $a \in U$.

Output: $A \in K_a$.

Method:
MSA(G, a)
BEGIN
Test:=true;
WHILE test AND there is an attribute $b \in U$ such that

$$\{b\} \rightarrow \{a\} \in S^+$$

DO BEGIN
$A := \{b\}$;
Test:=false
END
RETURN(A)
END.

Lemma 20. $A \in K_a$.

Proof. Because $\{a\} \in K_a$ and U is a finite set of attributes, the lemma is clear. □

The following lemma is obvious

Lemma 21. *The worst-case time complexity of MSA is* $\mathcal{O}(|U|^2|S|)$.

Remark 22. By Lemma 10 we have $A \to B \in S^+$ if and only if $\{a\} \to B \in S^+$ for every $a \in A$.

From this, we obtain the following lemma

Lemma 23. *Let* $G = (U, S)$ *be a strong scheme,* $a \in U, K_a$ *be a family of minimal sets of* $a, L \subseteq K_a, \{a\} \in L$. *Then* $L \subset K_a$ *if and only if there are* $C \in L, A \to B \in S^+$ *such that* $\forall E \in L \Rightarrow E \not\subseteq A \cup (C - B)$.

Proof. Suppose that $L \subset K_a$. Hence, there exists a $D \in K_a - L$. By $\{a\} \in L$ and the definition of K_a, we have

$$D \to \{a\} \in S^+ \tag{2}$$

and

$$a \notin D. \tag{3}$$

If for every SD $A \to B \in S$ implies $(A \cap D \neq \emptyset, B \subseteq D)$, or $A \cap D = \emptyset$, then $D^+ = D$. Therefore, by (3) we have $D \to \{a\} \notin S^+$. Which contradicts (2). Hence, there exists a SD $A \to B \in S$ such that $A \subseteq D$ and $B \not\subseteq D$. From this and Remark 22 we have a C such that $C \in L, A \subseteq D$ and $C - B \subseteq D$. Clearly, $A \cup (C - B) \subseteq D$. Consequently, we obtain $E \not\subseteq A \cup (C - B)$ for every $E \in L$.

Conversely, assume that there are $C \in L, A \to B \in S^+$ such that

$$E \not\subseteq A \cup (C - B) \tag{4}$$

for every $E \in L$. By the definition of L we have $A \cup (C - B) \to \{a\} \in S^+$. Because $\{a\} \in L$, there is a D such that $D \in K_a, a \notin D$ and $D \subseteq A \cup (C - B)$. From (4) we obtain $E \not\subseteq D$ for all $E \in L$, i.e. $D \in K_a - L$, or $L \subset K_a$.

The lemma is proved. □

From this lemma and MSA we construct the following algorithm by induction

Algorithm 24. FAMMSA

Input: a strong scheme $G = (U, S)$ and $a \in U$.

Output: K_a.

Method:

Step 1. Set $L(1) = E(1) = \{\{a\}\}$.

Step i+1. If there are C and $A \to B$ such that $C \in L(i), A \to B \in S, \forall E \in L(i) \Rightarrow E \not\subseteq A \cup (C - B)$, then by MSA construct an $E(i+1)$, where $E(i+1) \subseteq A \cup (C - B)$ and $E(i+1) \in K_a$. We set

$$L(i+1) = L(i) \cup E(i+1).$$

In the converse case we set $K_a = L(i)$.

By Lemma 23 there exists a natural number n such that $K_a = L(n)$.
The following lemma is obvious

Lemma 25. *The worst-case time complexity of FAMMSA is*

$$\mathcal{O}(|U|^2|S||K_a|(1 + |U||K_a|)).$$

By (3) in Proposition 18 we are easy to see that the time complexity of FAMMSA is polynomial in $|U|$ and $|S|$. Consequently, our algorithm is very effective.

It is obvious that if $S = \{\{a\} \to B_i : i = 1, \ldots, m\}$ or for each SD $A \to B \in S^+$ implies $a \notin B$, then $K_a = \{\{a\}\}$.

Let $G = (U, S)$ be a strong scheme over U. Set

$$MAX(S^+, a) = \{A \subseteq U : (A \to \{a\} \notin S^+) \text{ and } ((A \subset B) \Rightarrow (\exists D \subset B)(D \to \{a\} \in S^+)\}.$$

It can be seen that

$$MAX(S^+, a) = K_a^{-1} \quad \forall a \in U. \tag{5}$$

Denote $MAX(S^+) = \bigcup_{a \in U} MAX(S^+, a)$.

Lemma 26. *If* $U - \cup MAX(S^+) \neq \emptyset$ *then*

$$\{c\} \to U \in S^+,$$

where for every $c \in U - \cup MAX(S^+)$.

Proof. Suppose that $c \in U - \cup MAX(S^+)$. Hence $c \notin \cup MAX(S^+)$. By (5) we have

$$\{c\} \notin K_a^{-1} \quad \forall a \in U.$$

According to Proposition 18 and the definition of set of antikeys we have

$$\{c\} \in K_a \quad \forall a \in U.$$

Consequently by (S5) in Definition 3 and the definition of K_a we immediately get

$$\{c\} \to U \in S^+.$$

The lemma is proved. □

Lemma 27. *For every* $b \in A, A \in K_a^{-1} : \{b\} \to \{c\} \notin S^+$, *where* $c \in U - A$.

Proof. Assume that there exists an $A \in K_a^{-1}$ and $b \in A$ such that $\{b\} \to \{c\} \in S^+$. Because $A \in K_a^{-1}$ and $c \in U - (A \cup \{a\})$, we have $\{c\} \in K_a$. Then by Proposition 18 we have

$$\{c\} \to \{a\} \in S^+, \quad a \in U.$$

Hence, by (S2) in Definition 3 we obtain

$$\{b\} \to \{a\} \in S^+.$$

Which contradicts the facts that $A \in K_a^{-1}$ and $b \in A$. Therefore, we have $\{b\} \to \{c\} \notin S^+ \forall b \in A, A \in K_a^{-1}$ and $c \in U - (A \cup \{a\})$.

The lemma is proved. □

Now we assume that $MAX(S^+) = \{A_1, \ldots, A_t\}$. Then we defined the mapping $Max : U \longrightarrow \mathcal{P}(U)$ as follows:

$$Max(a) = \begin{cases} \bigcap\limits_{a \in A_i} A_i & \text{if } \exists A_i \in MAX(S^+) : a \in A_i, \\ U & \text{otherwise.} \end{cases}$$

It is easy to see that $\forall a \in U : a \in Max(a)$, and hence $Max(a) \neq \emptyset$. On the other hand, we are easy to see that if $S = \{\{a_1\} \to U, \ldots, \{a_n\} \to U\}$ where $U = \{a_1, \ldots, a_n\}$ then

$$\forall a_i \in U : \quad Max(a_i) = U.$$

Lemma 28. *If* $Max(a) = \{a\} \cup A, A \neq \emptyset$ *and* $a \notin A$ *then* $\{a\} \to A \in S^+$.

Proof. First we suppose that there is $b \in A$ such that $\{a\} \to \{b\} \notin S^+$. By Proposition 18 we get $\{a\} \notin K_b$. Assume that $K_b^{-1} = \{\{a\} \cup B\}$. It is clear that $\{b\} \in K_b$. Hence $b \notin \cup K_b^{-1}$, i.e. $b \notin B$. It can be seen that if $B \neq \emptyset$ then $A \subseteq B$. Thus we obtain $b \in B$. This is a contradiction. Therefore, $B = \emptyset$ holds. By the definition of $Max(a)$ we obtain $Max(a) = \{a\}$. Which conflicts with the fact that $Max(a) = \{a\} \cup A, A \neq \emptyset$ and $a \notin A$. Consequently, we have

$$\{a\} \to \{b\} \in S^+ \quad \forall b \in A.$$

From this and according to (S5) in Definition 3 we immediately get

$$\{a\} \to A \in S^+.$$

The Lemma is proved. □

By Lemma 28 it is obvious that if $Max(a) = U$ then $\{a\} \to U \in S^+$.

The following theorem gives a necessary and sufficient condition for an arbitrary relation to be Armstrong relation of a strong scheme.

Theorem 29. *Let $G = (U, S)$ be a strong scheme, $r = \{h_1, \ldots, h_m\}$ a relation over U. Then a necessary and sufficient condition for r to be Armstrong relation of strong scheme G is*

$$\forall a \in U : \{a\}_r^+ = Max(a),$$

where $\{a\}_r^+ = \{b \in U : \{a\} \to \{b\} \in S_r\}$.

Proof. First we show that $\{a\}^+ = Max(a)$ for all $a \in U$. Denote $H = \{A_i : A_i \in MAX(S^+)$ and $a \in A_i\}$. It can be seen that if $H = \emptyset$ then according to Lemma 26 we get $\{a\} \to U \in S^+$.

Suppose that $H \neq \emptyset$. It is easy to see that if $H \subseteq MAX(S^+)$ holds then by Lemma 28 we have $\{a\} \to Max(a) \in S^+$.

By Lemma 27, it is obvious that for any M such that $M \supset Max(a)$ we have $\{a\} \to M \notin S^+$.

Consequently, according to the definition of $\{a\}^+$ we have

$$\forall a \in U : \{a\}^+ = Max(a). \tag{6}$$

Obviously, according to Theorem 8 we can see that $S_r = S^+$ iff for every $a \in U : \{a\}^+ = \{a\}_r^+$ holds. Hence, if $S_r = S^+$ holds then $\{a\}_r^+ = Max(a)$ for all $a \in U$.

Conversely, we suppose that $\{a\}_r^+ = Max(a)$ for all $a \in U$. Then by Theorem 8 and (6) we obtain $S_r = S^+$.

The theorem is proved. □

Now we construct an algorithm that from a given strong scheme G finds a relation r such that r is Armstrong relation of G.

Algorithm 30.

Input: a strong scheme $G = (U, S)$.

Output: a relation r such that $S_r = S^+$.

Method:

Step 1. By FAMMSA compute K_a for each $a \in U$.

Step 2. By Algorithm 14 we compute K_a^{-1} for each $a \in U$.

Step 3. Set

$$MAX(S^+) = \bigcup_{a \in U} K_a^{-1}.$$

Step 4. Denote elements of $MAX(S^+)$ by $A_1, \ldots, A_t$. We construct a relation $r = \{h_0, h_1, \ldots, h_t\}$ as follows

$$\text{for all } a \in U, \quad h_0(a) = 0, \quad \forall i = 1, \ldots, t$$

$$h_i(a) = \begin{cases} 0 & \text{if } a \in A_i, \\ i & \text{otherwise.} \end{cases}$$

By Theorem 29 we have r is an Armstrong relation of G, i.e. $S_r = S^+$.

The following example shows that for a given strong scheme G, Algorithm 30 can be applied to construct a relation r such that r is an Armstrong relation of G.

Example 31. A strong scheme $G = (U, S)$, where $U = \{a, b, c, d\}$ and $S = \{\{a, b\} \to \{c\}, \{b\} \to \{a, d\}, \{d\} \to \{b\}\}$.

Then we have

$K_a = \{\{a\}, \{b\}, \{d\}\}, K_b = \{\{b\}, \{d\}\}, K_c = \{\{a\}, \{b\}, \{c\}, \{d\}\}, K_d = \{\{b\}, \{d\}\}$.

$K_a^{-1} = \{\{c\}\}, K_b^{-1} = \{\{a, c\}\}, K_c^{-1} = \emptyset, K_d^{-1} = \{\{a, c\}\}$.

$MAX(S^+) = \{\{a, c\}, \{c\}\}$.

Consequently

$$r = \begin{array}{cccc} a & b & c & d \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 2 & 2 & 0 & 2 \end{array}$$

It is obvious that $S_r = S^+$.

Algorithm 32. [8]

Input: a Sperner-system $K_{a_i} = \{B_1, \ldots, B_{m_i}\}$ over U.

Output: $K_{a_i}^{-1}$.

Method:

Step 1. We set $K_{i_1} = \{U - \{a\} : a \in B_1\}$. It is clear that $K_{i_1} = \{B_1\}^{-1}$.

Step q+1 $(q < m_i)$. We suppose that $K_{i_q} = F_{i_q} \cup \{X_1, \ldots, X_{t_{i_q}}\}$, where $X_1, \ldots, X_{t_{i_q}}$ containing B_{q+1} and $F_{i_q} = \{A : A \in K_{i_q}, B_{q+1} \not\subseteq A\}$. For all j $(j = 1, \ldots, t_{i_q})$ we construct the antikeys of $\{B_{q+1}\}$ on X_j in an analogous way as K_{i_1}. Denote them by $A_1^j, \ldots, A_{r_i}^j$ $(j = 1, \ldots, t_{i_q})$. Let

$$K_{i_{q+1}} = F_{i_q} \cup \{A_p^j : A \in F_{i_q} \Rightarrow A_p^j \not\subset A, 1 \le j \le t_{i_q}, 1 \le p \le r_j\}.$$

We set $K_{a_i}^{-1} = K_{i_m}$.

Denote $K_{i_q} = F_{i_q} \cup \{X_1, \ldots, X_{t_{i_q}}\}$ and $l_{i_q} (1 \le q \le m_i - 1)$ be the number of elements of K_{i_q}.

It is easy to see that the time complexity of Algorithm 30 is the time complexity of step 1 and step 2. By Proposition 16 and Lemma 25, the following proposition is clear.

Proposition 33. *The worst-case time complexity of Algorithm 30 is*

$$\mathcal{O}(n^2 \sum_{i=1}^{n} (\sum_{q=1}^{m_i - 1} t_{i_q} u_{i_q} + |S| m_i (1 + n m_i)))$$

where

$$U = \{a_1, \ldots, a_n\}, m_i = |K_{a_i}|,$$

$$u_{i_q} = \begin{cases} l_{i_q} - t_{i_q} & \text{if } l_{i_q} > t_{i_q}, \\ 1 & \text{if } l_{i_q} = t_{i_q}. \end{cases}$$

In the cases for which $l_{i_q} \leq l_{m_i}$ $(\forall i, \forall q : 1 \leq q \leq m_i)$, it is easy to see that the time complexity of our algorithm is

$$\mathcal{O}(n^2 \sum_{i=1}^{n} |K_{a_i}|(|S| + n|K_{a_i}||S| + |K_{a_i}^{-1}|^2)).$$

By (3) and (4) in Proposition 18 we are easy to see that the time complexity of Algorithm 30 is polynomial in $|U|$ and $|S|$. Consequently, our algorithm is very effective.

References

[1] Armstrong W. W., *Dependency structure of database relationship*, Information Processing 74, North-Holland Pub. Co. , (1974) 580-583.

[2] Czédli G., *Dependencies in the relational model of data (Hungarian)*, Alkalmaz Mat. Lapok **6** (1980), 131-143.

[3] Czédli G., *On dependencies in the relational model of data*, EIK **17** (1981), 103-112.

[4] Demetrovics J.,*Logical and structural investigation of relation datamodel*, MTA SZTAKI Tanulm-ányok **114** (1980), 1-97 (in Hungarian).

[5] Demetrovics J., Gyepesi G., *On the functional dependency and generalizations of it.* Acta Cybernetica Hungary **3** (1983), 295-305.

[6] Demetrovics J.,Thi V. D., *Armstrong relations, functional dependencies and strong dependencies*, Computers and Artificial Intelligence **14** (1995), 279-298.

[7] Thi V. D., *Strong dependencies and s-semilattices,* Acta Cybernetica **8** (1987), 195-202.

[8] Thi V. D., *Minimal keys and Antikeys*, Acta Cybernetica **7** (1986), 361-371.

[9] Thi V. D., Son N. H., *Some problems related to keys and the Boyce-Codd normal form*, Acta Cybernetica **16, 3** (2004), 473-483.

Received April, 2005

Acta Cybernetica **17** (2006) 533–555.

Demonic Fixed Points

Fairouz Tchier*

Abstract

We deal with a relational model for the demonic semantics of programs. The demonic semantics of a while loop is given as a fixed point of a function involving the demonic operators. This motivates us to investigate the fixed points of these functions. We give the expression of the greatest fixed point with respect to the demonic ordering (*demonic inclusion*) of the semantic function. We prove that this greatest fixed coincides with the least fixed point with respect to the usual ordering (*angelic inclusion*) of the same function. This is followed by an example of application.

Keywords: demonic fixed points, demonic functions, while loops, relational demonic semantics.

1 Introduction

We use relations to define the input-output semantics of nondeterministic programs. The relational operators $\cup$ and ; have been used for many years to define the so-called *angelic semantics*, which assumes that a program goes right when there is a possibility to go right. On the other hand, the demonic operators $\sqcup$ and $\sqcap$ (to be introduced below) do the opposite: if there is a possibility to go wrong, a program whose semantics is given by these operators goes wrong; it is the *demonic semantics* of nondeterministic programs.

The concept of fixed points has received increasing attention from researchers in a wide range of scientific areas, especially in computer science. Many important notions in demonic relational semantics are given as fixed points of some monotonic functions involving the demonic operators [4, 16, 25, 39, 44]. In this paper, we concentrate on while loops. We will present some relevant results about the fixed points of the function $f(X) = Q \sqcap P \sqcap X$ (these operators will be defined later). By considering certain conditions on the domains of P and Q, one gets the demonic semantics we have assigned to while loops in previous papers [22, 47, 48, 49, 50]. Other similar definitions of while loops can be found in [28, 38, 43, 51, 52, 53]. Our results can be applied also to the program verification and construction; while there

*Mathematics department, King Saud University, P.O.Box 22452, Riyadh 11495, Saudi Arabia, E-mail: ftchier@hotmail.com

is no systematic way to calculate the relational abstraction of a while loop directly from the definition it is possible to check the correctness of any candidate abstraction by theorem 30. For deterministic programs, Mills has described a checking method [34, 35].

The approach to demonic input-output relation presented here is not the only possible one. In [28, 29, 30], the infinite looping has been treated by adding to the state space a fictitious state $\perp$ to denote nontermination. In [9, 24, 32, 40], the demonic input-output relation is given as a pair (relation,set). The relation describes the input-output behavior of the program, whereas the set component represents the domain of guaranteed termination. In [19, 17, 18, 31], they abstract from relational semantics to the setting of modal Kleene algebras, an extension of Kozen's Kleene algebra with tests.

We note that the preponderant formalism employed until now for the description of demonic input-output relation is the wp-calculus. For more details see [1, 3, 23].

The rest of the paper is organized as follows. In Section 2, we present our mathematical tool, namely relation algebra [13, 42, 45]. First, we recall the basic laws (Subsection 2.1). In Section 3, we present notions related to fixed points followed by a description of our refinement ordering (Section 4) and finally in Section 5, we give our main results. In Section 6, we give an application.

2 Relation algebras

2.1 Definition and basic laws

Our mathematical tool is abstract relation algebra [13, 42, 45], which we now introduce.

Definition 1. *A* (homogeneous) relation algebra *is a structure* $(\mathcal{R}, \cup, \cap, \bar{}, \breve{}, ;)$ *over a non-empty set* $\mathcal{R}$ *of elements, called* relations. *The following conditions are satisfied.*

- $(\mathcal{R}, \cup, \cap, \bar{})$ *is a complete Boolean algebra, with* zero *element* $\emptyset$, universal *element* L *and ordering* $\subseteq$.
- Composition, *denoted by* $(;)$, *is associative and has an identity element, denoted by* I.
- *The Schröder rule is satisfied:* $P;Q \subseteq R \Leftrightarrow \breve{P};\overline{R} \subseteq \overline{Q} \Leftrightarrow \overline{R};\breve{Q} \subseteq \overline{P}$.
- $L;R;L = L \Leftrightarrow R \neq \emptyset$ *(Tarski rule).*

The relation $\breve{R}$ is called the *converse* of R. The standard model of the above axioms is the set $\wp(S \times S)$ of all subsets of $S \times S$. In this model, $\cup, \cap, \bar{}$ are the usual *union, intersection* and *complement*, respectively; the relation $\emptyset$ is the empty relation, the universal relation is $L = S \times S$ and the identity relation is $I = \{(s, s') \mid s' = s\}$. Converse and composition are defined by

$$\breve{R} = \{(s, s') \mid (s', s) \in R\} \quad \text{and} \quad Q;R = \{(s, s') \mid \exists s'' : (s, s'') \in Q \wedge (s'', s') \in R\}.$$

The precedence of the relational operators from highest to lowest is the following: $\overline{\ }$ and $\breve{\ }$ bind equally, followed by ;, then by $\cap$, and finally by $\cup$. From now on, the composition operator symbol ; will be omitted (that is, we write QR for $Q;R$). From Definition 1, the usual rules of the calculus of relations can be derived (see, e.g., [9, 13, 42]). We assume these rules to be known and simply recall a few of them.

Theorem 2. *Let P, Q, R be relations. Then,*

(a) $\overline{Q \cup R} = \overline{Q} \cap \overline{R}$,

(b) $\overline{Q \cap R} = \overline{Q} \cup \overline{R}$,

(c) $Q \cap R \cup \overline{R} = Q \cup \overline{R}$,

(d) $P \cap Q \subseteq R \Leftrightarrow P \subseteq \overline{Q} \cup R$,

(e) $Q \subseteq R \Leftrightarrow \overline{R} \subseteq \overline{Q}$,

(f) $P(Q \cap R) \subseteq PQ \cap PR$,

(g) $(P \cap Q)R \subseteq PR \cap QR$,

(h) $P(Q \cup R) = PQ \cup PR$,

(i) $(P \cup Q)R = PR \cup QR$,

(j) $Q \subseteq R \Rightarrow PQ \subseteq PR$,

(k) $Q \subseteq R \Rightarrow QP \subseteq RP$.

(l) $\overline{RL}L = \overline{RL}$,

(m) $PQ \cap R \subseteq P(Q \cap \breve{P}R)$,

(n) $(P \cap QL)R = PR \cap QL$,

(o) $(\bigcap_{i \in X} R_i L)L = \bigcap_{i \in X} R_i L$.

We now give a definition of various properties of relations.

Definition 3. *A relation R is* functional *iff $\breve{R}R \subseteq I$. A relation v is a vector [42] iff $v = vL$.*

In the standard model, a relation R on a set S is functional iff $(s, s') \in R \wedge (s, s'') \in R \Rightarrow s' = s''$. A vector is a relation of the form $T \times S$, where $T \subseteq S$. A vector can also be viewed as a point set or a predicate. For example, if $S = \{0, 1, 2\}$ and $T = \{0, 1\}$, then $t := T \times S = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2)\}$ is a vector that corresponds to the point set T. For any relation R, the relation RL is a vector that characterizes the domain of R. For instance, with $R := \{(0,1), (0,2), (2,1)\}$ (on set $S = \{0,1,2\}$), we obtain $RL = \{(0,0), (0,1), (0,2), (2,0), (2,1), (2,2)\}$; this vector indeed characterizes the domain of R, which is $\{0, 2\}$.

2.2 Relative implication

In our work we need to define an operator called *relative implication.* In previous work, we used the monotype and residual operators see [49, 47, 48]

Definition 4. *A binary operator $\triangleleft$, called* relative implication *[50], is defined as follows :*

$$Q \triangleleft R := \overline{Q\overline{R}}.$$

This operator has a dual operator (Definition 10), $\triangleright$*, given by :*

$$Q \triangleright R := \overline{\overline{Q}R}.$$

The most interesting case is when the right argument is a vector RL, in other words $Q \triangleleft RL$. If $x.R$ denotes the set of the images of x by R, then $x \in \mathrm{dom}(Q \triangleleft RL) \Leftrightarrow x.Q \subseteq \mathrm{dom}(R)$.

The operators $\triangleleft$ and $\triangleright$ bind equally but less than (;) and more than $\cap$ and $\cup$. In the next lemma we give some interesting properties satisfied by the operator $\triangleleft$. The properties of $\triangleright$ can be obtained by dualization of those of the operator $\triangleleft$ [50].

Lemma 5. *Let P, Q and R be relations.*

(a) $\overline{Q \triangleleft R} = Q\overline{R}$,

(b) $\overline{Q \triangleright R} = \overline{Q}R$,

(c) $P \triangleleft Q \cap P \triangleleft R = P \triangleleft (Q \cap R)$,

(d) $P \triangleleft R \cap Q \triangleleft R = (P \cup Q) \triangleleft R$,

(e) $PQ \triangleleft R = P \triangleleft (Q \triangleleft R)$,

(f) $PQ \cap P \triangleleft R = P(Q \cup \overline{R}) \cap P \triangleleft R$,

(g) $P \triangleleft Q \subseteq PQ \cup \overline{PL}$,

(h) $P \subseteq Q \Rightarrow Q \triangleleft R \subseteq P \triangleleft R$,

(i) $P \subseteq Q \Rightarrow R \triangleleft P \subseteq R \triangleleft Q$

(j) Q *vector* $\Rightarrow P \triangleleft Q$ *vector.*

We note that the properties (c) and (e) are similar to those of the logical operators $\rightarrow$, $\wedge$ and $\vee$. For example, the property (e) corresponds to $(P \wedge Q \rightarrow R) \leftrightarrow (P \rightarrow (Q \rightarrow R))$.

3 Fixed points

Let f be a monotonic function with respect to $\subseteq$. The least fixed point of f is $\bigcap\{X \mid f(X) = X\}$. Similarly, $\bigcup\{X \mid f(X) = X\}$ is the greatest fixed point of f. Because we assume our relation algebra to be complete (Definition 1), least and greatest fixed points of monotonic functions exist. We will denote the least fixed point of the function $f(X) := E(x)$, where E is some relational expression, by μf or by $\mu(X : E(X))$, when it is desired not to introduce a function name. Similarly,

νf and $\nu(X : E(X))$ denote the greatest fixed point of f. The following properties of fixed points are used below:

$$\begin{array}{lll} \text{(a)} & \mu f = \bigcap\{X | f(X) = X\} = \bigcap\{X | f(X) \subseteq X\}, & (1) \\ \text{(b)} & \nu f = \bigcup\{X | f(X) = X\} = \bigcup\{X | X \subseteq f(X)\}, & \\ \text{(c)} & \mu f \subseteq \nu f, & \\ \text{(d)} & f(Y) \subseteq Y \Rightarrow \mu f \subseteq Y, & \\ \text{(e)} & Y \subseteq f(Y) \Rightarrow Y \subseteq \nu f. & \end{array}$$

We need some auxiliary notions.

Definition 6. *A function f between complete lattices is* strict *if $f(\emptyset) = \emptyset$ and* co-strict *if $f(L) = L$. Further, f is called* continuous *if for every chain C one has $f(\bigcup C) = \bigcup f(C)$, and* co-continuous *if for every chain C one has $f(\bigcap C) = \bigcup f(C)$.*

Every universally disjunctive function is continuous and strict; every universally conjunctive function is co-continuous and co-strict. Moreover, every continuous or co-continuous function is monotonic.

Theorem 7. *[44] (Knaster-Tarski) Every monotonic endofunction on a complete lattice has a least fixed point μf, which is equal to its least pre-fixed point and a greatest fixed point $\nu(f)$ which is equal to its greatest post-fixed point. If f is continuous, then $\mu(f) = \bigcup\{f^i(\emptyset) : i \in \mathbb{N}\}$. If f is co-continuous, then $\nu(f) = \bigcap\{f^i(L) : i \in \mathbb{N}\}$.*

Here,

$$\begin{array}{rcl} f^0(x) & = & x\,, \\ f^{i+1}(x) & = & f(f^i(x))\,. \end{array}$$

For more details about these notions see [12, 15]. The comparison of fixed points is sometimes very useful to compare the program semantics. The next proposition will present some results in this direction.

Proposition 8. *Let $(X, \leq)$ be an ordered set f and g endofunctions. Let also the relation $\ll$ on the set of endofunctions on X, defined as follows :*

$$f \ll g \Leftrightarrow (\forall x : x \in X : f(x) \leq g(x)).$$

We have the following properties ($+$ is a monotonic binary operation on X) :

$$\begin{array}{lll} \textit{(a)} & \mu \textit{ monotonic} & f \ll g \Rightarrow \mu(f) \leq \mu(g), \\ \textit{(b)} & \textit{permutation law} & \mu(f\,;g) = f(\mu(g\,;f)), \\ \textit{(c)} & \textit{diagonal law} & \mu(x \mapsto x + x) = \mu(x \mapsto \mu(y \mapsto x + y)), \\ \textit{(d)} & \mu\textit{-fusion law} & f\,;g = g\,;h \Rightarrow \mu(f) = g(\mu(h)), \\ & & (g \textit{ is continuous and strict}). \end{array}$$

3.1 Transitive reflexive closure

Another operation that occurs in the definition of the while program semantics is the *reflexive transitive closure.* The *reflexive transitive closure* is a unary operation denoted $*$ and defined for every relation R by :

$$R^* = \mu(X \mapsto I \cup RX). \tag{2}$$

The operation $*$ is defined as a least fixed point ; by the monotonicity of $\cup$ and of $(;)$, and the Knaster-Tarski Theorem (7) this operation is well defined and it satisfies,

$$R^* = I \cup RR^* = I \cup R^*R. \tag{3}$$

The unary operations $*, \breve{}$ and $^-$ bind equally. We can also define a similar operation to $*$, called *transitive closure*, denoted $+$, and defined for every relation R by :

$$R^+ = \mu(X \mapsto R \cup RX). \tag{4}$$

and,

$$\begin{array}{ll} \text{(a)} & R^+ = R^*R = RR^* = R \cup RR^*, \\ \text{(b)} & R^* = I \cup R^+. \end{array} \tag{5}$$

The operations $*$ and $+$ bind equally. The operation $*$ satisfies also

$$R^* = \bigcup_{i \geq 0} R^i, \tag{6}$$

where $R^0 = I$ and $R^{i+1} = RR^i$.

We give some properties of the operation $*$. The properties of the operation $+$ are easily deduced from the equations 5 and of the properties of $*$.

$$R^*Q = \mu(X \mapsto Q \cup RX), \tag{7}$$

Proposition 9. *Let P, Q and R be relations. We have,*

- $PQP = PQ \Rightarrow (PQ)^*P = PQ^*$,
- $QR = \emptyset \Rightarrow (Q \cup R)^* = R^*Q^*$,
- $RQ = \emptyset$ *and* $QR = \emptyset \Rightarrow (R \cup Q)^* = R^* \cup Q^*$,

We need the notion of *dual* function.

Definition 10. *Let f be an endofunction on a Boolean lattice. The* dual *function of f is $f^\#(x) := \overline{f(\overline{x})}$.*

The next Lemma states the relationship between the fixed points of a function on a Boolean lattice and those of its dual function.

Lemma 11. *Let f be an endofunction on a Boolean lattice and $f^{\#}$ its dual function. If x is a fixed point of f, then*

(a) $\overline{x}$ *is a fixed point of* $f^{\#}$,

(b) $\nu(f^{\#}) = \overline{\mu(f)}$,

(c) $\mu(f^{\#}) = \overline{\nu(f)}$.

3.2 The initial part of a relation

In the sequel, we describe notions that are useful for the description of the set of initial states of a program for which termination is guaranteed. These notions are the *initial part* of a relation and *progressive finiteness*. The *initial part* of a relation R, denoted $\mathcal{L}(R)$, is the vector characterizing the set of points s_0 such that there is no infinite chain $s_0, s_1, s_2, \ldots$, with $(s_i, s_{i+1}) \in R$, for all $i \geq 0$. The algebraic definition is

Definition 12. *[42] The* initial part *of a relation R, denoted $\mathcal{L}(R)$, is given by :*

$$\mathcal{L}(R) := \bigcap\{x \mid R \triangleleft x = x\},$$

where x takes its value in the set of the vectors (by Theorem 2(o), $\mathcal{L}(R)$ is a vector). (see [41, 42]); in other words, $\mathcal{L}(R)$ is the least fixed point of the $\subseteq$-monotonic function $g(x) := R \triangleleft x$, where x is a vector (the least fixed point of g exists since the set of vectors is also a complete lattice [42]). A relation R is said to be progressively finite iff $\mathcal{L}(R) = L$, in other words if there is no infinite path by R. Progressive finiteness of a relation R is the same as well-foundedness of $\breve{R}$. (Mnemonics : $\mathcal{L}$ for loop because, in the program semantics, $\mathcal{L}(R)$ represents the set of states from which no infinite loop is possible.)

We find in [50] an equivalent definition of $\mathcal{L}(R)$ on the set of relations instead of vectors. This definition is given in the next proposition

Proposition 13. *Let R be a relation.*

$$\begin{aligned} (a)\quad \mathcal{L}(R) &= \bigcap\{X \mid R \triangleleft X = X\} \\ &= \bigcap\{X \mid R \triangleleft X \subseteq X\} \\ &= \bigcap\{X \mid R\overline{X} = \overline{X}\} \\ &= \mu(X \mapsto R \triangleleft X) \end{aligned}$$

and

$$\begin{aligned} (b)\quad \overline{\mathcal{L}(R)} &= \bigcup\{X \mid X = RX\} \\ &= \bigcup\{X \mid X \subseteq RX\} \\ &= \nu(X \mapsto RX). \end{aligned}$$

Proof. These results can be easily deduced from the Equations 1, 13(a), of the definition of $\triangleleft$ and certain Boolean laws. □

Let us give the formal definition of a progressively finite relation.

Definition 14. *A relation R is called* progressively finite *[42] iff $\mathcal{L}(R) = L$. In an omega algebra [14] the complement of the initial part is known as the* infinite iteration.

By using the results of Proposition 13, we have :

$$R \text{ is progressively finite } \Leftrightarrow \overline{\mathcal{L}(R)} = \emptyset \Leftrightarrow (\forall X : X \subseteq RX \Rightarrow X = \emptyset). \quad (8)$$

The next proposition presents some properties of the initial part of a relation [50]. The properties (a), (b) and (c) can be found also in [42].

Proposition 15. *Let Q and R be relations.*

(a) $\mathcal{L}(R) \subseteq R^*\overline{RL}$,

(b) $\bigcup_{i\geq 0} \overline{R^i L} \subseteq \mathcal{L}(R)$,

(c) R *deterministic* $\Rightarrow \mathcal{L}(R) = R^*\overline{RL}$,

(d) $Q \subseteq R \Rightarrow \mathcal{L}(R) \subseteq \mathcal{L}(Q)$,

(e) $R \triangleleft \mathcal{L}(R) = \mathcal{L}(R)$ *(equivalent to* $R\overline{\mathcal{L}(R)} = \overline{\mathcal{L}(R)}$*)*,

(f) $\mathcal{L}(R) = \mathcal{L}(R^+)$,

(g) $R^* \triangleleft \mathcal{L}(R) = R^+ \triangleleft \mathcal{L}(R) = \mathcal{L}(R)$ *(equivalent to* $R^*\overline{\mathcal{L}(R)} = R^+\overline{\mathcal{L}(R)} = \overline{\mathcal{L}(R)}$*)*,

(h) Q *progressively finite* $\Rightarrow Q \cap R$ *progressively finite,*

(i) $R \cap \mathcal{L}(R)$ *is progressively finite.*

Proof. See [50].

4 A demonic refinement ordering

We now define the refinement ordering we will be using in the sequel. This ordering induces a complete join semilattice, called a *demonic semilattice.* The associated operations are demonic join ($\sqcup$), demonic meet ($\sqcap$) and demonic composition ($\diamond$). We give the definitions and needed properties of these operations. For more details on relational demonic semantics and demonic operators, see [6, 7, 8, 9, 21, 22, 50].

Definition 16. *We say that a relation Q refines a relation R [33], denoted by $Q \sqsubseteq R$, iff*

$$Q \cap RL \subseteq R \ \wedge \ RL \subseteq QL.$$

Thus, for instance,

$$\begin{pmatrix}1&0&1\\1&1&0\\1&0&0\end{pmatrix} \sqsubseteq \begin{pmatrix}1&0&1\\0&0&0\\1&0&0\end{pmatrix}, \text{ but } \begin{pmatrix}1&0\\0&0\end{pmatrix} \not\sqsubseteq \begin{pmatrix}1&1\\1&0\end{pmatrix}$$

(these Boolean matrices represent relations over sets by the well-known correspondence).

Proposition 17. *Let Q and R be relations. We have,*

(a) The greatest lower bound (wrt $\sqsubseteq$) of relations Q and R is

$$Q \sqcup R = (Q \cup R) \cap QL \cap RL.$$

And here is an example of this operation:

$$\begin{pmatrix}1&1&0\\1&0&0\\0&0&0\end{pmatrix} \sqcup \begin{pmatrix}0&0&0\\0&1&0\\0&1&1\end{pmatrix} = \begin{pmatrix}0&0&0\\1&1&0\\0&0&0\end{pmatrix}.$$

This operation corresponds to a demonic non-deterministic choice, since the possibility of failure (row 3 of the first matrix or row 1 of the second) is reflected in the result. For the middle row, failure is not possible, and the set of allowed results is the union of the results of the two operands.

(b) If Q and R satisfy the condition $QL \cap RL = (Q \cap R)L$, their least upper bound is

$$Q \sqcap R = (Q \cap R) \cup \overline{QL} \cap R \cup Q \cap \overline{RL},$$

otherwise, the least upper bound does not exist see [11, 22].

The existence condition simply means that on the intersection of their domains, Q and R have to agree for at least one value.

Figure 1 shows the general structure of $(\mathcal{A}_{\cup R}, \sqsubseteq)$, for a given R. It is shown in [22] that it is a complete join semilattice. Let f be a monotonic function (wrt $\sqsubseteq$) having at least one fixed point. Because $(\mathcal{A}_{\cup R}, \sqsubseteq)$ is a complete join semilattice, the following properties of fixed points can be transferred from Equations 1.

$$\begin{array}{ll}(a) & \nu f \;=\; \bigsqcup\{X | f(X) = X\} \;=\; \bigsqcup\{X | X \sqsubseteq f(X)\}, \\ (b) & Y \sqsubseteq f(Y) \;\Rightarrow\; Y \sqsubseteq \nu f.\end{array} \tag{9}$$

In the sequel we will introduce some operation, related to the usual relational composition, the so-called *demonic composition*. Its definition is

Definition 18. $Q \sqsupset R := QR \cap Q \lhd RL$.

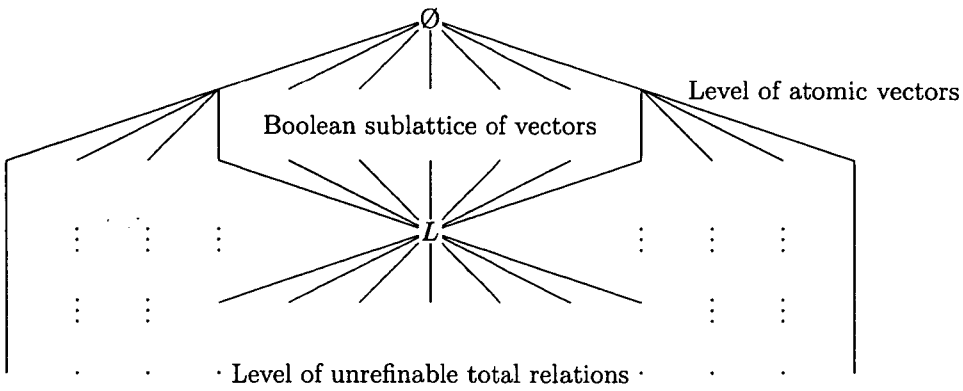


Figure 1: General structure of a semilattice ordered by $\sqsubseteq$.

A pair (s,t) belongs to $Q \,\square\, R$ if and only if it belongs to QR and there is no possibility of reaching, from s, by Q, an element u that does not belong to the domain of R. For example, if $Q = \{(0,0),(0,1),(1,2)\}$ and $R = \{(0,0),(2,3)\}$, one finds that $Q \,\square\, R = \{(1,3)\}$; the pair $(0,0)$, which belongs to QR, does not belong to $Q \,\square\, R$, since $(0,1) \in Q$ and 1 is not in the domain of R. Note that we assign to $\square$ the same binding power as that of ;.

Proposition 19.

(a) Q *deterministic* $\Rightarrow Q \,\square\, R = QR$,

(b) P *deterministic* $\Rightarrow P \,\square\, (Q \sqcap R) = PQ \sqcap PR$,

(c) R *total* $\Rightarrow Q \,\square\, R = QR$,

(d) $PL \cap QL = \emptyset \Rightarrow P \sqcap Q = P \cup Q$,

(e) $PL \cap QL = \emptyset \Rightarrow P \sqcup Q = \emptyset$,

(f) $PL \cap QL = \emptyset \Rightarrow (P \cup Q) \,\square\, R = P \,\square\, R \cup Q \,\square\, R$,

(g) $P \sqsubseteq Q \;\wedge\; R \sqsubseteq S \;\wedge$
$(P \cap RL) \cup (R \cap SL) \subseteq (Q \cap PL) \cup (S \cap RL) \Rightarrow P \cup R \sqsubseteq Q \cup S$.

The next proposition demonstrates a number of additional properties. Of particular interest is item (c), which shows that demonic composition distributes on the right over intersection when one of the intersected entities is a vector.

Lemma 20. *Let* Q, R *be relations and* u, v *vectors. We have,*

(a) $(v \cap Q) \,\square\, R = v \cap Q \,\square\, R$,

(b) $R \,\square\, v = RL \cap R \triangleleft v$,

(c) $Q \,\square\, (v \cap R) = Q \,\square\, v \cap Q \,\square\, R$, *[26, 43]*

(d) $Q \sqsubseteq R \Leftrightarrow v \cap Q \sqsubseteq v \cap R \wedge \overline{v} \cap Q \sqsubseteq \overline{v} \cap R$,

(e) $u \sqsubseteq v \;\Rightarrow\; P \lhd u \cap Q \sqsubseteq P \lhd v \cap Q$,

(f) $R \sqsubseteq v \cap R$,

(g) $v \cap Q \sqsubseteq R \;\Rightarrow\; Q \sqsubseteq R$.

5 Demonic Fixed points

During the execution of a program in an input state, by considering a demonic point of view (if there is a possibility for the program not to terminate normally then it will not terminate normally), three cases may happen: normal termination, abnormal termination and infinite loops. In this section, we will give formally the input-output relation of a program and show that is equal to the greatest fixed point of the the semantic function of the while loop **do** P $\rightarrow$ Q **od**. We will give a few facts about fixed points.

The next theorem highlights the importance of progressive finiteness in the simplification of fixed point-related properties.

Theorem 21. *[5] Let* $f(X) := Q \cup PX$ *be a function. If* P *is progressively finite, the function* f *has a unique fixed point which means that* $\nu(f) = \mu(f) = P^*Q$.

Because YL is a vector characterizing the domain of relation Y, the following theorem qualifies the range of domains of fixed points of f. We note that in the case when the relation P is progressively finite, we find the results of Theorem 21.

Theorem 22. *Every fixed point* Y *of* $f(X) := Q \cup PX$ *satifies*

$$P^*Q \subseteq Y \subseteq P^*Q \cup \overline{\mathcal{L}(P)}$$

and P^*Q *and* $P^*Q \cup \overline{\mathcal{L}(P)}$ *are respectively the least and the greatest fixed point of the function* f.

For proof see [5]. The next corollary is about the fixed points of the function $g(X) := Q \cap P \lhd X$.

Corollary 23. *Every fixed point* Y *of* $g(X) := Q \cap P \lhd X$ *satisfies*

$$P^* \lhd Q \cap \mathcal{L}(P) \subseteq Y \subseteq P^* \lhd Q$$

$P^* \lhd Q \cap \mathcal{L}(P)$ *and* $P^* \lhd Q$ *are respectively the least and the greatest fixed points of the function* g.

Proof. It is easy to verify that g is the dual function (Definition 10) of $f(X) := \overline{Q} \cup PX$. By Lemma 11, $\overline{Y}$ is a fixed point of f. By Theorem 22, $\overline{Y}$ verifies

$$P^*\overline{Q} \subseteq \overline{Y} \subseteq P^*\overline{Q} \cup \overline{\mathcal{L}(P)}.$$

By applying DeMorgan Laws, this is equivalent to

$$\overline{P^*\overline{Q}} \cap \mathcal{L}(P) \subseteq Y \subseteq \overline{P^*\overline{Q}}.$$

Finally, by Definition 4,

$$P^* \triangleleft Q \cap \mathcal{L}(P) \subseteq Y \subseteq P^* \triangleleft Q.$$

We note that, if the relation P is progressively finite, the function g has a unique fixed point which is $P^* \triangleleft Q$. □

We introduce the next Abbreviations:

Abbreviation 24. Let P and Q be relations. The Abbreviations d, d_L and $\mathcal{A}(P,Q)$ are defined as follows (x is a vector) :

$$\begin{aligned} d(X) &:= (Q \cup PX) \cap P \triangleleft XL, \\ d_L(x) &:= (PL \cup QL) \cap P \triangleleft x, \\ \mathcal{A}(P,Q) &:= P^* \triangleleft (PL \cup QL) \\ \mathcal{S} &:= P^*Q \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P). \end{aligned}$$

(Mnemonics : the subscript L refers to the fact that d_L is obtained from d by composition with L; $\mathcal{A}$ stands for *a*bnormal, since it represents states from which abnormal termination is not possible; finally, $\mathcal{S}$ stands for *s*emantics, since it represents states from which no infinite loop is possible.

5.1 Intuition

- The function d can be considered as a generalization of the semantic function of the while loop **do** P $\rightarrow$ Q **od**.
- In a nondeterministic loop **while** P **do** Q, P is iteratively applied to a state s until Q holds. As, P is nondeterministic s can have many outputs. If among these outputs there is a state which can lead outside the domain of P or of Q, so this state is excluded (abnormal termination of the loop). So, $\mathcal{A}(P,Q)$ represents the states from which no abnormal termination is possible.
- We note that $\mathcal{S}$ is an intersection of three terms; P^*Q, $\mathcal{A}(P,Q)$ and $\mathcal{L}(P)$. By taking in consideration the intuition behind these terms, it is easy to see that the relation $\mathcal{S}$ represents the set of states from which the termination is guaranteed because all the states from which there is a possibility of nontermination (abortion or infinite loop) are excluded by the terms $\mathcal{A}(P,Q)$ and $\mathcal{L}(P)$.

The following lemma presents the relationship between the fixed points of the functions d and d_L (Abbreviation 24).

Lemma 25. *If Y is a fixed point of d then YL is a fixed point of d_L.*

Proof. Suppose that $d(Y) = Y$.

$$
\begin{array}{ll}
 & d_L(YL) \\
= & \quad \langle \text{ Definition of } d_L \text{ (24). } \rangle \\
 & (PL \cup QL) \cap P \triangleleft YL \\
= & \quad \langle \ L = YL \cup \overline{YL}. \ \rangle \\
 & (QL \cup P(YL \cup \overline{YL}) \cap P \triangleleft YL \\
= & \quad \langle \text{ Boolean laws, Theorem 2(i) and Lemma 5(f) . } \rangle \\
 & (Q \cup PY)L \cap P \triangleleft YL \\
= & \quad \langle \text{ Lemma 5(j) and Theorem 2(n). } \rangle \\
 & ((Q \cup PY) \cap P \triangleleft YL)L \\
= & \quad \langle \text{ Definition of } d \text{ (24) and } d(Y) = Y. \ \rangle \\
 & YL
\end{array}
$$

□

In the sequel we give the bounds of the fixed points of d_L and show that, these bounds are also fixed points of d_L.

Theorem 26. *If Y is a fixed point of d, then*

(a) $\mathcal{A}(P,Q) \cap \mathcal{L}(P) \subseteq YL \subseteq \mathcal{A}(P,Q)$,

(b) $\mathcal{A}(P,Q) \cap \mathcal{L}(P)$ *and* $\mathcal{A}(P,Q)$ *are fixed points of* d_L.

Proof.

1. By Lemma 25, YL is a fixed point of d_L. By taking $Q := PL \cup QL$ in Corollary 23, we find,

$$P^* \triangleleft (PL \cup QL) \cap \mathcal{L}(P) \subseteq YL \subseteq P^* \triangleleft (PL \cup QL);$$

by using Abbreviation 24, we find

$$\mathcal{A}(P,Q) \cap \mathcal{L}(P) \subseteq YL \subseteq \mathcal{A}(P,Q).$$

2. By Corollary 23, we deduce that $\mathcal{A}(P,Q) \cap \mathcal{L}(P)$ and $\mathcal{A}(P,Q)$ are fixed points of d_L.

□

The next theorem characterizes the domain of $\mathcal{S}$ (24). This domain is the set of points for which normal termination is guaranteed (no possibility of abnormal termination or infinite loop).

Theorem 27. *Let $\mathcal{S}$ given by the Abbreviation 24. We have*

$$\mathcal{S}L = \mathcal{A}(P,Q) \cap \mathcal{L}(P).$$

Proof. $\mathcal{S}L$

$=$ ⟨ Abbreviation 24. ⟩

$(P^*Q \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P))L$

$=$ ⟨ Theorem 2(n). ⟩

$P^*QL \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P)$

$=$ ⟨ Abbreviation 24 and Lemma 5(f). ⟩

$(P^*QL \cup P^*\overline{PL \cup QL}) \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P)$

$=$ ⟨ Theorem 2(b,h). ⟩

$P^*(QL \cup \overline{PL} \cap \overline{QL}) \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P)$

$=$ ⟨ Theorem 2(c). ⟩

$P^*(QL \cup \overline{PL}) \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P)$

$=$ ⟨ Theorem 2(h). ⟩

$P^*QL \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P) \cup P^*\overline{PL} \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P)$

$=$ ⟨ Proposition 15(a) and Boolean Law. ⟩

$\mathcal{A}(P,Q) \cap \mathcal{L}(P)$

□

In what follows, we will present some interesting properties satisfied by $\mathcal{S}$ and relations that have the same domain as $\mathcal{S}$.

Lemma 28. *Let R be a relation and $\mathcal{S}$ given by Abbreviation 24. Then*

$$R \,\square\, \mathcal{S} = RP^*Q \cap R \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P)),$$

Proof. $R \,\square\, \mathcal{S}$

$=$ ⟨ Definition 18 and Theorem 27. ⟩

$R\mathcal{S} \cap R \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P))$

$=$ ⟨ Abbreviation 24, Theorem 2(c,h), Lemma 5(f) and Boolean law. ⟩

$R(P^*Q \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P) \cup \overline{\mathcal{A}(P,Q) \cap \mathcal{L}(P)}) \cap R \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P))$

$=$ ⟨ Theorem 2(c,h) and Lemma 5(f). ⟩

$RP^*Q \cap R \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P))$

□

In the following theorem, we will show that $\mathcal{S}$ is a fixed point of d (Abbreviation 24 and demonic composition 18). The function d can be also given by,

$$d(X) = Q \cap P \triangleleft XL \cup P \,\square\, X \tag{10}$$

Theorem 29. *$\mathcal{S}$ (Abbreviation 24) is a fixed point of d.*

Proof. $d(\mathcal{S})$

$= \quad$ ⟨ Abbreviation 24 and Theorem 27. ⟩

$(Q \cup P\mathcal{S}) \cap (P \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P)))$

$= \quad$ ⟨ Theorem 2(h). ⟩

$Q \cap (P \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P))) \cup P\mathcal{S} \cap (P \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P)))$

$= \quad$ ⟨ $Q \subseteq QL \cup PL$ and Theorem 27. ⟩

$Q \cap (PL \cup QL) \cap (P \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P))) \cup P\mathcal{S} \cap P \triangleleft \mathcal{S}L$

$= \quad$ ⟨ Definition 18, Abbreviation 24 and Theorem 26(b). ⟩

$Q \cap (\mathcal{A}(P,Q) \cap \mathcal{L}(P)) \cup P \,\square\, \mathcal{S}$

$= \quad$ ⟨ Lemma 28. ⟩

$Q \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P) \cup PP^*Q \cap P \triangleleft \mathcal{A}(P,Q) \cap \mathcal{L}(P)$

$= \quad$ ⟨ $PP^*Q \subseteq PL \cup QL$, Theorem 26(b) and Boolean law. ⟩

$(Q \cup PP^*Q) \cap (\mathcal{A}(P,Q) \cap \mathcal{L}(P))$

$= \quad$ ⟨ Theorem 2(i) and Equation 3. ⟩

$P^*Q \cap (\mathcal{A}(P,Q) \cap \mathcal{L}(P))$

$= \quad$ ⟨ Abbreviation 24. ⟩

$\mathcal{S}$

□

The following theorem is a generalization to a nondeterministic context of the *while statement verification rule* of Mills [34, 35]. It shows that the least fixed point W of d (Abbreviation 24) is uniquely characterized by conditions (a) and (b), that is, by the fact that W is a fixed point of d and by the fact that no infinite loop is possible when the execution is started in a state that belongs to the domain of W. Half of this theorem (the $\Leftarrow$ direction) is also proved by Sekerinski (the *main iteration theorem* [43]) in a predicative programming setup.

Theorem 30. *W is the least fixed point wrt $\subseteq$ of d (Abbreviation 24) ($W = \mu_{\subseteq}(d)$) iff*

$$\begin{array}{ll} (a) & W = d(W), \\ (b) & WL \subseteq \mathcal{L}(P). \end{array}$$

Proof. ($\Rightarrow$) : As W is the least fixed point of d then, (a) is evident. Since $W = \mu(d) \subseteq \mathcal{S}$, then $WL = \mu(d)L \subseteq \mathcal{S}L$, by using Theorem 27, we have $WL \subseteq \mathcal{L}(P)$.

($\Leftarrow$) : By Hypothesis (a), W is a fixed point of d. Then, by Theorem 26, $\mathcal{A}(P,Q) \cap \mathcal{L}(P) \subseteq WL \subseteq \mathcal{A}(P,Q)$. But, by using Hypothesis (b), $WL \subseteq \mathcal{L}(P)$, then $WL = \mathcal{A}(P,Q) \cap \mathcal{L}(P)$.

$$
\begin{aligned}
&= && \langle \text{ Hypothesis } \rangle \\
&d(W) \\
&= && \langle \text{ Abbreviation24. } \rangle \\
&(Q \cup PW) \cap P \triangleleft WL \\
&= && \langle\ (Q \cup PW) \subseteq (PL \cup QL) \text{ and } WL = \mathcal{A}(P,Q) \cap \mathcal{L}(P)\ \rangle \\
&(Q \cup PW) \cap (PL \cup QL) \cap P \triangleleft (\mathcal{A}(P,Q) \cap \mathcal{L}(P,Q)) \\
&= && \langle \text{ Lemma 28(b) } \rangle \\
&(Q \cup PW) \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P) \\
&= && \langle \text{ Theorem 2(n)and Boolean law. } \rangle \\
&Q \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P) \cup (P \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P))W
\end{aligned}
$$

So W is a fixed point of the function $g(X) := Q \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P) \cup (P \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P))X$. Since, by Proposition 15(i,h), $P \cap \mathcal{A}(P,Q) \cap \mathcal{L}(P)$ is progressively finite, invoking Theorem 21 shows that g has a unique fixed point which is the least fixed point $\mu(d)$. We conclude that $W = \mu(d)$. □

The next theorem shows that $\mathcal{S}$ is the least fixed point of d wrt $\subseteq$ $(\mathcal{S} = \mu_{\subseteq}(d))$.

Theorem 31. *$\mathcal{S}$ (Abbreviation24) is the least fixed point of d wrt $\subseteq$ $(\mathcal{S} = \mu_{\subseteq}(d))$.*

Proof. It suffices to prove that $\mathcal{S}$ satisfies the conditions of the Theorem 30. By Theorem 29, $\mathcal{S}$ is a fixed point of d. So, the condition (a) is satisfied. By Theorem 27, $\mathcal{S}L = \mathcal{A}(P,Q) \cap \mathcal{L}(P)$, so that condition (b) holds too. □

In the sequel, we show that $\mathcal{S}$ is the greatest fixed point with respect to $\sqsubseteq$ of d (Equation 10). In other words, we will show that the least fixed point of d wrt $\subseteq$ is equal to the greatest fixed point of the same function d wrt $\sqsubseteq$. But before we have to prove that d is monotonic wrt $\sqsubseteq$.

Lemma 32. *The function d (Abbreviation 24) is monotonic wrt $\sqsubseteq$.*

Proof. Let X and Y be relations such that $X \sqsubseteq Y$.

As $\square$ is monotonic, we have $P \square X \sqsubseteq P \square Y$ and $XL \sqsubseteq YL$. By Proposition 20(e), we have $Q \cap P \triangleleft XL \sqsubseteq Q \cap P \triangleleft YL$.

By taking in Proposition 19(g) $P := P \square X \wedge Q := P \square Y \wedge R := Q \cap P \triangleleft XL \wedge \mathcal{S} := R := Q \cap P \triangleleft YL$, it is easy to see that the conditions are satisfied, whence we conclude that $d(X) \sqsubseteq d(Y)$. Thus, d is monotonic wrt $\sqsubseteq$ □

As the function d is monotonic (wrt $\sqsubseteq$)(Lemma 32), by Theorem 7 and Equation 9(a) the greatest fixed point $\mathcal{S}$ of d exists and is given by

$$\mathcal{S} = \bigsqcup\{X | X = Q \cap P \triangleleft XL \cup P \square X\}; \tag{11}$$

(Since $\mathcal{S}$ is a fixed point of d (Theorem 29) and thus $\bigsqcup\{X | X = Q \cap P \triangleleft XL \cup P \square X\}$ is well defined, by completness of the $\bigsqcup$-semilattice; for more information about the properties of the $\bigsqcup$-semilattice see [22]).

In the next theorem, we show that $\mathcal{S}$ is the greatest fixed point with respect to $\sqsubseteq$ of d (Equation 10).

Theorem 33. *$\mathcal{S}$ is the greatest fixed point with respect to $\sqsubseteq$ of d ($\mathcal{S} = \nu_{\sqsubseteq}(d)$).*

Proof. Let $W = \nu_{\sqsubseteq}(d)$. Let's show that $W = \mu_{\subseteq}(d)$. Then $\mathcal{S} \sqsubseteq W$, by Definition 16 and Theorem 26, $WL \subseteq \mathcal{S}L \subseteq \mathcal{L}(P)$. Also, we have $W = d(W)$. By Theorem 30, $W = \mu_{\subseteq}(d)$. Hence $W = \mathcal{S}$. □

The following example is an application of our results. It is rather contrived, but it is simple and fully illustrate the various cases that may happen.

6 Application

In [7, 8], Berghammer and Schmidt propose abstract relation algebra as a practical means for the specification of data types and programs. Often, in these specifications, a relation is characterized as a fixed point of some function. Can demonic operators be used in the definition of such a function? Let us now show with a simple example that the concepts presented in this paper give useful insights for answering this question.

In [7, 8], it is shown that the natural numbers can be characterized by the relations z and S (*zero* and *successor*) and the laws

$$\begin{array}{lll} \text{(a)} & \emptyset \neq z = zL \ \wedge \ z\check{z} \subseteq I & (z \text{ is a point}), \\ \text{(b)} & S\check{S} = I \ \wedge \ \check{S}S \subseteq I & (S \text{ is a one to one function.}), \\ \text{(c)} & Sz = \emptyset & (z \text{ has no predecessor}), \\ \text{(d)} & L = \bigcap\{x | z \cup \check{S}x = x\} & (\text{generation principle}). \end{array} \tag{12}$$

By Proposition 19(a,c) these equations imply,

$$\begin{array}{lll} \text{(a)} & \emptyset \neq z = z \sqcap L \ \wedge \ z \sqcap \check{z} \subseteq I & (z \text{ is a } \textit{demonic} \text{ point}), \\ \text{(b)} & S \sqcap \check{S} = I \ \wedge \ \check{S} \sqcap S \subseteq I & (S \text{ is a one to one function.}), \\ \text{(c)} & S \sqcap z = \emptyset & (z \text{ has no } \textit{demonic} \text{ predecessor}), \\ \text{(d)} & L = \bigcap\{x | z \cup \check{S} \sqcap x = x\} & (\textit{demonic} \text{ generation principle}). \end{array} \tag{13}$$

By Proposition 19(a), Theorem 22 and the last axiom, the function

$$g(X) = z \cup \check{S} \sqcap X \tag{14}$$

obviously has a unique solution for X, namely, $X = L$ and this solution is expressed by

$$L = \check{S}^* z \tag{15}$$

However, it is easy to show that $z \sqcup \check{S} \sqcap X \subseteq X$, obtained by replacing the join in Equation 14 by its demonic counterpart, has infinitely many solutions. Indeed, from $S \sqcap z = \emptyset$, Proposition 19(c) and the Schröder rule, it follows that $z \cap \check{S} \sqcap L = \emptyset$, so by Proposition 19(e), we have $z \sqcup \check{S} \sqcap X = \emptyset$.

Hence, any relation R is a solution to $z \sqcup \breve{S} \sqcap X \subseteq X$. Looking at Figure 1, one immediately sees why it is impossible to reach L by joining anything to z (which is a point and hence is an immediate $\sqsubseteq$-predecessor of $\emptyset$), since this can only lead to z or to $\emptyset$.

Let us now go 'fully demonic' and ask what is a solution to $z \sqcup \breve{S} \sqcap X \sqsubseteq X$[1]. By the discussion above, this is equivalent to $\emptyset \sqsubseteq X$, which has a unique solution, $X = \emptyset$. This raises the question whether it is possible to find some fully demonic function similar to (14), whose fixed point is $X = L$. Because L is in the middle of the demonic semilattice (Figure 1), there are in fact two possibilities: either approach L from above or from below.

For the approach from above, consider the function

$$h(X) := z \sqcap \breve{S} \sqcap X. \tag{16}$$

In other words, we have to find the post-fixed point of this function (wrt $\sqsubseteq$). By Proposition 19(d), we have, $z \sqcap \breve{S}X = z \cup \breve{S}X$. This means that 16 reduces to

$$h(X) := z \cup \breve{S}X. \tag{17}$$

By Equation 15 and Theorem 33, L is the greatest fixed point of k (wrt $\sqsubseteq$); so $L = \bigsqcup\{X | X \sqsubseteq z \sqcap \breve{S} \sqcap X\}$.

Now consider $\bigsqcap_{n\geq 0} \breve{S}^{\underline{n}} \sqcap z$, where $\breve{S}^{\underline{n}}$ is an n-fold demonic composition defined by $\breve{S}^{\underline{0}} = I$ and $\breve{S}^{\underline{n+1}} = \breve{S} \sqcap \breve{S}^{\underline{n}}$. By axiom 12(b), $\breve{S}$ is deterministic, so that, by 19(a) and associativity of demonic composition, $\breve{S}^{\underline{n}} \sqcap z = \breve{S}^n z$. In the sequel, we will prove that $\bigsqcap_{n\geq 0} \breve{S}^{\underline{n}} \sqcap z$ is defined and is equal to L.

First,

$$\begin{array}{ll} & \bigsqcap_{n\geq 0} \breve{S}^{\underline{n}} \sqcap z \text{ defined} \\ \Leftrightarrow & \bigsqcap_{n\geq 0} \breve{S}^n z \text{ defined} \\ \Leftrightarrow & \qquad \langle \text{ Proposition 17(a). } \rangle \\ & L \subseteq (\bigcap_{n\geq 0} \breve{S}^n z \cup \overline{\breve{S}^n z L}) L \\ \Leftrightarrow & \qquad \langle \text{ Axiom 12(a), } zL = z \text{ and thus } \breve{S}^n z \cup \overline{\breve{S}^n z L} = L. \rangle \\ & \text{true.} \end{array}$$

Second,

$$\begin{array}{ll} & \bigsqcap_{n\geq 0} \breve{S}^{\underline{n}} \sqcap z \\ = & \bigsqcap_{n\geq 0} \breve{S}^n z \\ = & \qquad \langle \text{ Proposition 17(b). } \rangle \\ & (\bigcap_{n\geq 0} \breve{S}^n \cup \overline{\breve{S}^n z L}) \cap (\bigcup_{n\geq 0} \breve{S}^n z L) \end{array}$$

[1]This inequation contains the conversion operator, which is not a demonic operator. However, it could be suppressed by using relation $P := \breve{S}$ as a basic constant in specification 12, rather than using S.

$$
\begin{array}{ll}
= & \langle \text{ Axiom 12(a), } zL = z \text{ and thus } \breve{S}^n z \cup \overline{\breve{S}^n z L} = L. \rangle \\
& \bigcup_{n\geq 0} \breve{S}^n z \\
= & \langle \text{ 15. } \rangle \\
& L.
\end{array}
$$

□

So, $\bigsqcap_{n\geq 0} \breve{S}^{\underline{n}} \sqcap z = \bigsqcup\{X | X \sqsubseteq z \sqcap \breve{S} \sqcap X\}$.

It is easy to show that for any $n \geq 0$, $\breve{S}^n z$ is a point (it is the n-th successor of zero) and that $m \neq n \Rightarrow \breve{S}^m z \neq \breve{S}^n z$. Hence, in $(\mathcal{R}_L, \sqsubseteq)$, $\{\breve{S}^n z | n \geq 0\}$ (i.e. $\{\breve{S}^{\underline{n}} \sqcap z | n \geq 0\}$) is the set of immediate predecessors of $\emptyset$; looking at Figure 1 shows how the universal relation L arises as the greatest lower bound $\bigsqcap_{n\geq 0} \breve{S}^{\underline{n}} \sqcap z$ of this set of points.

Note that, whereas there is a unique solution to 14, there are infinitely many solutions to 16 (equivalently, to 17), for example $\bigsqcup_{n\geq h} S^{\underline{n}}$ $(= \bigcup_{n\geq k} S^n)$, for any k.

For the upward approach, consider

$$k(X) := \check{z} \sqcup X \sqcap S. \tag{18}$$

We will find the pre-fixed point of k (wrt $\sqsubseteq$) Here again there are infinitely many solutions to this case; in particular, any vector v, including $\emptyset$ and L, is a solution to 18. Because $(\mathcal{R}, \sqsubseteq)$ is only a join semilattice, it is not at all obvious that the least fixed point of $k(X) := \check{z} \sqcup X \sqcap S$ exists. It does, however, since the following derivation shows that $\bigsqcup_{n\geq 0} \check{z} \sqcap S^{\underline{n}}$ $(= \bigsqcup_{n\geq 0} k^n(\check{z})$, where $k^0(\check{z}) = \check{z})$ is a fixed point of k and hence is obviously the least solution of 18:

$$
\begin{array}{ll}
& \check{z} \sqcup (\bigsqcup_{n\geq 0} \check{z} \sqcap S^{\underline{n}}) \sqcap S \\
= & \langle \text{ Associativity of } \sqcap, \text{ and } S \sqcap I = SI = S. \rangle \\
& \check{z} \sqcap I \sqcup (\bigsqcup_{n\geq 0} \check{z} \sqcap S^{\underline{n+1}}) \\
= & \langle \ S^{\underline{0}} = I. \ \rangle \\
& \check{z} \sqcap S^{\underline{0}} \sqcup (\bigsqcup_{n\geq 1} \check{z} \sqcap S^{\underline{n}}) \\
= & \bigsqcup_{n\geq 0} \check{z} \sqcap S^{\underline{n}}.
\end{array}
$$

Because $\check{z}$ and S are functions, Proposition 19(a) implies that $\check{z} \sqcap S^{\underline{n}} = \check{z} S^n$, for any $n \geq 0$. But $\check{z} S^n$ is also a function (it is the inverse of the point $\breve{S}^n z$) and hence is total, from which, by Proposition 17(a), law 2(l) and equation 15,

$$\bigsqcup_{n\geq 0} \check{z} \sqcap S^{\underline{n}} = \bigsqcup_{n\geq 0} \check{z} S^n = \bigcup_{n\geq 0} \check{z} S^n = (\bigcup_{n\geq 0} \breve{S}^n z)^{\smile} = \breve{L} = L.$$

This means that L is the least upper bound of the set of mappings $\{\check{z} \sqcap S^{\underline{n}} | n \geq 0\}$. Again, a look at Figure 1 gives some intuition to understand this result, after recalling that mappings are minimal elements in $(\mathcal{R}_L, \sqsubseteq)$ (though not all mappings have the form $\check{z} \sqcap S^{\underline{n}}$).

Thus, building L from below using the set of mappings $\{\check{z} \sqcap S^{\underline{n}} | n \geq 0\}$ is symmetric to building it from above using the set of points $\{\breve{S}^{\underline{n}} \sqcap z | n \geq 0\}$. □

Acknowledgement

We thank the anonymous referee for his or her comments.

References

[1] R. J. R. Back. On the correctness of refinement in program development. Thesis, Department of Computer Science, University of Helsinki, 1978.

[2] R. J. R. Back. On correct refinement of programs. *J. Comput. System Sci. 23*, 1, 1981, 49–68.

[3] R. J. R. Back and J. von Wright. Combining angels, demons and miracles in program specifications. *Theoret. Comput. Sci. 100*, 1992, 365–383.

[4] R. C. Backhouse et al. Fixed points calculus. *Inform. Pro. Letters,53*, 1995, 131–136.

[5] R. C. Backhouse and H. Doornbos. Mathematical induction made calculational. Computing Science Note 94/16, Dept. of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, 1994.

[6] R. C. Backhouse and J. van der Woude. Demonic operators and monotype factors. *Mathematical Structures in Comput. Sci. 3*, 4, 417–433, 1993.

[7] R. Berghammer. Relational specification of data types and programs. Technical report 9109, Fakultät für Informatik, Universität der Bundeswehr München, Germany, 1991.

[8] R. Berghammer and G. Schmidt. Relational specifications. In C. Rauszer, editor, *Algebraic Logic*, volume 28 of *Banach Center Publications*. Polish Academy of Sciences, 1993.

[9] R. Berghammer and H. Zierer. Relational Algebraic semantics of deterministic and nondeterministic programs. *Theoret. Comput. Sci. 43*, 123–147, 1986.

[10] R. Bird and O. de Moor. *Algebra of programming*. Prentice Hall, London, 1997.

[11] N. Boudriga, F. Elloumi and A. Mili. On the lattice of specifications: Applications to a specification methodology. *Formal Aspects of Computing 4*, 1992, 544–571.

[12] C. Brink, W. Kahl, and G. Schmidt, editors. *Relational Methods in Computer Science*. Springer, 1997.

[13] L. H. Chin and A. Tarski. Distributive and modular laws in the arithmetic of relation algebras. *University of California Publications 1*, 1951, 341–384.

[14] E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Proc. of Mathematics of Program Construction, 5th International Conference,MPC 2000*,volume 1837 of *LNCS*, pages 45-59. Springer, 2000.

[15] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge Mathematical Textbooks, Cambridge University Press, Cambridge, 1990.

[16] J. Desharnais and B. Möller. Least Reflexive Points of Relations. To appear in *Higher Order and Symbolic Computation.*

[17] J. Desharnais, B. Möller, and G Struth. Termination in modal Kleene algebra. In J.-J. Levy, E Mayr, and J. Mitchell, editors, *Proc. IFIP TCS 2004*, pages 653-666. Kluwer, 2004.

[18] J. Desharnais, B. Möller, and G Struth. Applications of modal Kleene algebra. *Journal on Relational Methods in Computer Science 1:93–131 (2004).* http://www.cosc.brocku.ca/Faculty/Winter/JoRMiCS/

[19] J. Desharnais, B. Möller, and G Struth. Kleene algebra with domain. Technical Report 2003-07, Universität Augsburg, Institut für Informatik, June 2003. Revised version to appear in ACM Transactions on Computational Logic. http://www.informatik.uni-augsburg.de/forschung/techBerichte/reports/2003-7.pdf

[20] J. Desharnais and B. Möller. Characterizing determinacy in Kleene algebras. *Informations Sciences,* 139(3-4):253-273, December 2001.

[21] J. Desharnais, B. Möller, and F. Tchier. Kleene under a demonic star. *8th International Conference on Algebraic Methodology And Software Technology (AMAST 2000)*, May 2000, Iowa City, Iowa, USA, *Lecture Notes in Computer Science*, Vol. 1816, pages 355–370, Springer-Verlag, 2000.

[22] J. Desharnais, N. Belkhiter, S. B. M. Sghaier, F. Tchier, A. Jaoua, A. Mili and N. Zaguia. Embedding a demonic semilattice in a relation algebra. *Theoretical Computer Science*, 149(2):333–360, 1995.

[23] E. W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, N.J., 1976.

[24] H. Doornbos. A relational model of programs without the restriction to Egli-Milner monotone constructs. *IFIP Transactions*, A-56:363–382. North-Holland, 1994.

[25] M. Frappier, J. Desharnais, A. Mili and F. Mili. Verifying objects against axiomatic specifications: A fixed point approach. *5*th Conf. Maghrebian Conference on Software Engeneering and Artificial Intelligence, (MCSEAI'98). Pages 259-274, Tunis, Tunisie, December 1998.

[26] M. Frappier. A relational basis for program construction by parts. Dept. of Computer Science, University of Ottawa, 1994.

[27] E. Hehner. Predicative programming, Parts I and II. *Commun. ACM, 27*, February 1984, 134–151.

[28] C. A. R. Hoare and J. He. The weakest prespecification. *Fundamenta Informaticae IX*, 1986, Part I: 51–84, 1986.

[29] C. A. R. Hoare and J. He. The weakest prespecification. *Fundamenta Informaticae IX*, 1986, Part II: 217–252, 1986.

[30] C. A. R. Hoare and al. Laws of programming. *Communications of the ACM*, 30:672–686, 1986.

[31] Kleene algebras with tests. *ACM Transactions on Programming Languages and Systems*, 19, 427-443, 1997.

[32] R. D. Maddux. Relation-algebraic semantics. *Theoretical Computer Science*, 160:1–85, 1996.

[33] A. Mili, J. Desharnais and F. Mili. Relational heuristics for the design of deterministic programs. *Acta Inform. 24*, 3, 1987, 239–276.

[34] H. D. Mills. The new math of computer programming. *Commun. ACM 18*, 1, January 1975, 43–48.

[35] H. D. Mills, V. R. Basili, J. D. Gannon and R. G. Hamlet. *Principles of Computer Programming. A Mathematical Approach.* Allyn and Bacon, Inc., 1987.

[36] C. Morgan and K. Robinson. Specification statements and refinement. *IBM J. Res. Dev. 31*, 5, 1987. Reprinted in: C. Morgan and T. Vickers (eds). *On the refinement calculus.* Springer-Verlag, 1994, 23–46.

[37] J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Sci. Comput. Prog. 9*, 1987, 287–306.

[38] T. T. Nguyen. A relational model of demonic nondeterministic programs. *Int. J. Found. Comput. Sci. 2*, 2, 1991, 101–131.

[39] J. Cai and R. Paige. Program Derivation by Fixed point Computation. *Science of Computer Programming,*, 11,1989, 197–261.

[40] D. L. Parnas. A Generalized Control Structure and its Formal Definition. *Communications of the ACM*, 26:572–581, 1983

[41] G. Schmidt. Programs as partial graphs I: Flow equivalence and correctness. *Theoret. Comput. Sci. 15*, 1981, 1–25.

[42] G. Schmidt and T. Ströhlein. *Relations and Graphs.* EATCS Monographs in Computer Science, Springer-Verlag, Berlin, 1993.

[43] E. Sekerinski. A calculus for predicative programming. *Second International Conf. on the Mathematics of Program Construction.* R. S. Bird, C. C. Morgan and J. C. P. Woodcock (eds), Oxford, June 1992, *Lect. Notes in Comput. Sci. 669*, Springer-Verlag, 1993.

[44] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955, 285-309.

[45] A. Tarski. On the calculus of relations. *J. Symb. Log. 6*, 3, 1941, 73–89.

[46] F. Tchier. While loop demonic relational semantics monotype/residual style. *2003 International Conference on Software Engineering Research and Practice (SERP03)*, Las Vegas, Nevada, USA, 23-26, June 2003.

[47] F. Tchier. Demonic relational semantics monotype/residual style. *International Journal of Mathematics and Mathematical sciences*, 2003.

[48] F. Tchier. Demonic semantics by monotypes. *International Arab conference on Information Technology* (Acit2002), University of Qatar, Qatar, 16-19 December 2002.

[49] F. Tchier. Demonic relational semantics of compound diagrams. In: Jules Desharnais, Marc Frappier and Wendy MacCaull, editors. Relational Methods in computer Science: The Québec seminar, pages 117-140, Methods Publishers 2002.

[50] F. Tchier. Sémantiques relationnelles démoniaques et vérification de boucles non déterministes. Thése de doctorat, Département de Mathématiques et de statistique, Université Laval, Canada, 1996.
http://auguste.ift.ulaval.ca/∼ desharn/Theses/index.html

[51] M. Walicki and S. Meldal. Algebraic approches to nondeterminism: An overview. *ACM computing Surveys,29(1)*, 1997, 30-81.

[52] N. T. E. Ward. A refinement Calculus for Nondeterministic Expressions. Ph D thesis, University of Queensland, Australia, 1994.

[53] L.Xu, M. Takeichi and H. Iwasaki. Relational semantics for locally nondeterministic programs. *New Generation Computing 15*, 1997, 339-362.

Received March, 2004

Acta Cybernetica **17** (2006) 557–587.

Image reconstruction and correction methods in neutron and X-ray tomography*

Zoltán Kiss[†] Lajos Rodek[†] and Attila Kuba[†]

Abstract

Neutron and X-ray tomography are imaging techniques for getting information about the interior of objects in a non-destructive way. They reconstruct cross-sections from projection images of the object being investigated. Due to the properties of the image acquisition system, the projection images are distorted by several artifacts, and these reduce the quality of the reconstruction. In order to eliminate these harmful effects the projection images should be corrected before reconstruction. Taking projections is usually an expensive and time consuming procedure. One of our main goals has been to try to minimize the number of projections – for example, by exploiting more a priori information. A possible way of reducing the number of projections is by the application of discrete tomographic methods. In this case a special class of objects can be reconstructed, consisting of only a few homogenous materials that can be characterized by known discrete absorption values. To this end we have implemented two reconstruction methods. One is able to reconstruct objects consisting of cylinders and spheres made of homogeneous materials only. The other method is a general one in the sense that it can be used for reconstructing any shape. Simulations on phantoms and physical measurements were carried out and the results are presented here.

1 Introduction

Nowadays it is an interesting common task of physics and image processing to get information about the interior of an object without damaging it in any way. For this purpose several kinds of physical methods are deployed like X-ray, gamma, or neutron imaging. In industrial metallic specimen examination, neutron radiation is generally used. Tomography is an imaging tool for reconstructing objects from their projection images. (A brief review of the principles of tomography is given in Section 2.) However, the acquisition of neutron projections is a very time consuming

*This work was supported by the NSF grant DMS0306215 (Aspects of Discrete Tomography) and the OTKA grant T 048476 (New Aspects and Applications of Discrete Tomography in Neutron Radiography).

[†]Department of Image Processing and Computer Graphics, University of Szeged, H-6720 Szeged Árpád tér 2. E-mail: {kissz, rodek, kuba}@inf.u-szeged.hu

and expensive procedure, hence the goal is to do the reconstruction using as few projections as possible.

Discrete tomography (DT) is a special field of tomography where the object to be reconstructed consists of a small number of homogeneous regions with known absorption coefficients. For example, if the object is made of pure iron, the number of regions is 2 (iron and air) and the reconstructed function can have only two values: the absorption coefficients of iron and air. In DT we cleverly use the information that the function has a known discrete range. This is the main difference between the DT and the classical reconstruction techniques, as in the latter the function/object can in general have arbitrary (non-negative) values. (Comparison with other reconstruction techniques can be found in [7].) The knowledge of known, discrete absorption values may allow us, using DT methods, to reconstruct such objects from a small number of projections (e.g. 2-4) and/or to improve the quality of the reconstruction (more details about DT see [5]).

Two DT methods have been developed at our department. Both consider the reconstruction problem as an optimization task, but the main difference between the two methods lies in the object representation. One of them represents the object being investigated as a digital image and the other as a parameterized geometrical model. These two methods and results obtained using them in phantom/measured experiments are presented in Sections 5, 8 and 9, respectively. Both techniques have been integrated into the system called DIRECT (DIscrete REConstruction Techniques) [14], which is a framework available on-line that incorporates various DT methods being developed at our department.

Since the neutron images we get are usually distorted by several different effects, it is common practice to apply some correction methods before doing any reconstruction from the acquired projections. These effects are usually due to the incorrect settings of the image acquisition apparatus and the physical properties of the radiation used. These distorting effects and a description of the proposed correction techniques are given in Section 4, and our success in dealing with the former is outlined in Section 7.

2 Basic definitions

In this section we shall give a brief overview of the mathematical foundations of tomography, describing the methods, techniques, and algorithms used in this paper.

In non-destructive testing (NDT) several kinds of objects are imaged by some transmission rays like X-rays or neutron rays. The rays transmitted through the object are then partially absorbed. The relation between the initial (unabsorbed) and transmitted intensities, I_S and I_D respectively, can be expressed as a function that depends on the absorption coefficient (μ) of the object. Namely,

$$I_D(s,\vartheta) = I_S \cdot e^{-\int_s^D \mu(u)du}. \tag{1}$$

This equation is a basic relation in *transmission tomography*, where the cross-

sections of the object being studied are to be determined from such measurements. Mathematically the transmission tomography is modelled by the *Radon transformation*, giving the line integrals of a two-dimensional integrable function f, denoted by $\mathcal{R}f$. Formally,

$$[\mathcal{R}f](s,\vartheta) = \int_{-\infty}^{\infty} f(x,y)\,du\ , \tag{2}$$

where s and u denote the variables of the coordinate system rotated by ϑ. The function $\mathcal{R}f$ for a fixed value of ϑ is also called the ϑ-angle *projection* of f.

Let f denote the absorption coefficients of the 2D object being studied. Then the ϑ-angle projection of f can be computed from the transmission measurements after a suitable logarithmic transformation. That is

$$[\mathcal{R}f](s,\vartheta) = g(s,\vartheta) = \ln \frac{I_S}{I_D(s,\vartheta)}\ . \tag{3}$$

Then the *reconstruction problem* can be posed as one where the goal is to find a function f such that its projections are equal to some given function $g(s,\vartheta)$. In other words, we are looking for the inverse of $\mathcal{R}$, i.e., $\mathcal{R}^{-1}g$. The function f is sometimes called the *image function*, or briefly the *image*.

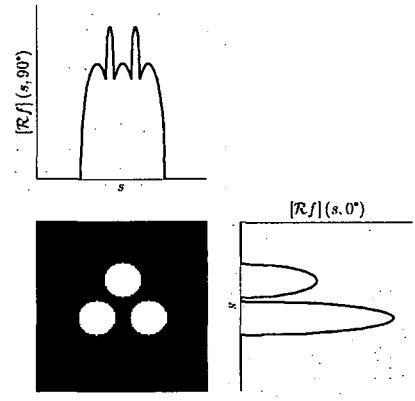


Figure 1: Horizontal and vertical projections of a binary image (black: 0, white: 1).

In this paper we are interested in a special kind of tomography called *discrete tomography*, where the range of the function to be reconstructed is a known discrete set. Such information is usually available in NDT, where the materials (and their absorption coefficients) of the object being studied are known. The simplest example of DT is when the range of the function f consists of only two values, like 0 and 1. In this case the image function is a *binary image* (see Fig. 1). A summary of the theory, algorithms, and applications of DT is given in [5].

3 Neutron and X-ray radiography

In order to get an image of an object, several kinds of radiation (gamma, neutron, X-ray) can be used. The principle of the apparatus of neutron radiography presented here is widely used nowadays, but it is employed in other imaging techniques as well (Fig. 2).

The object to be investigated is placed on a rotating table. The table can be rotated by a PC-controlled stepper motor, thus letting the beams pass through the object in different directions. The beams attenuated by the object impact on a scintillator, which then transforms the detected radiation into visible light detected by a CCD camera. Since the camera can be damaged by direct radiation, an optical mirror system conveys the light from the scintillator to the CCD camera. The images taken by the camera are temporarily stored by the camera controller, and finally a dedicated PC reads out the raw image data from it. A more thorough description of the imaging apparatus can be found in [1].

In this paper we will assume that the radiation source emits parallel beams. This assumption is not unrealistic since if the object being investigated is far enough from the reactor core, the transmitted beams will be almost parallel, so no significant geometric distortions will be introduced into the projections.

The remaining part of this section will discuss various artifacts often encountered during image acquisition.

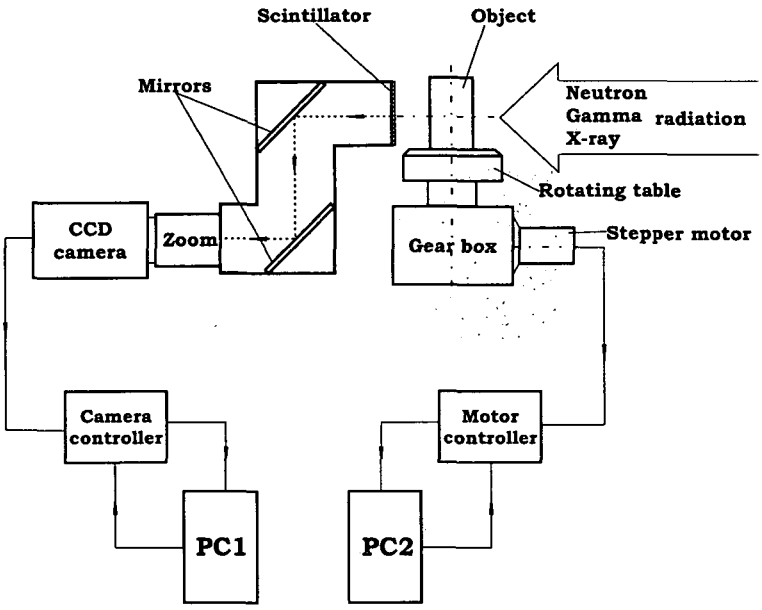


Figure 2: Apparatus for collecting projections.

3.1 Imaging artifacts

In neutron as well as in X-ray radiography the projection images can be distorted by several artifacts due to non-perfect imaging, errors of measurement, and errors in the model. These distortions should be corrected as best one can before or during the reconstruction, since reconstruction is a noise amplifying procedure.

Some of the distortions are due to the properties of the image acquisition system. For example, if the detector system is not uniformly sensitive in the whole field of view of projections, certain areas may be brighter, while others may be darker. This *nonuniformity* may cause ring artifacts in the reconstruction.

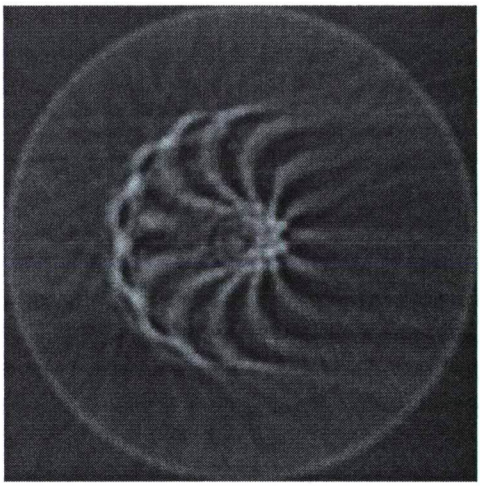


Figure 3: One of the reconstructed cross-sections of a Vidicon tube without applying the correction step.

A further problem might be when the *intensity* of the rays or the *sensitivity* of the camera changes during the acquisition period (cold or warm camera electronics may cause such effects, for instance). As a consequence of changing sensitivity, brighter or darker projections may be acquired and such images with artifacts may be reconstructed (see Fig. 3).

Another source of artifacts might be when the projections are taken not exactly as they should be in their necessary positions. For example, when the projection of the axis of rotation is not exactly in the centerline of the projection images. This *center of rotation* problem may blur the contours in the reconstructed images.

In our experience it is common for the projection images to randomly contain *white isolated points* owing to some problems with the detector system. For example, some pixels may be burnt out in the detector plane.

4 Pre-processing

The input data of the reconstruction procedure is the set of projection images taken from different directions. In neuton tomography these images are not suitable for direct reconstruction, as is shown in Section 3.1. Hence different pre-processing steps are needed prior to the reconstruction procedure. The pre-processing steps in our system are the following.

1. *Logarithmic transformation* In order to get an approximate values of the line integrals of the object along the transmitting rays, we first have to perform a logarithmic transformation on the measured intensity values, as given in (3).

2. *Cropping.* The projections of the object being investigated often cover only a small part of the whole acquired images, so the relevant part is selected and cropped from all projections (see Fig. 9). Only the cropped projections with a smaller image size are used for the further pre-processing steps and reconstruction. The reconstruction from cropped projections requires less memory and computational time.

3. *Motion correction.* It can also happen that the settings of the projection images were not perfect and that some of the images were not taken from the right position. A consequence might be that the images are the rotated or translated versions of the correct ones. In general, such distortions cannot be corrected if we do not know anything about the movement during acquisition or we have no information about the object to be reconstructed. Otherwise, there is some chance of correcting these artifacts. For example, if the data acquisition can be repeated (with the same or with some other object) and the same setting errors occurred, then the translations and rotations can be estimated and performed as a pre-processing step (e.g. 'center of rotation correction' [8]). In another case, if we know that the object is circularly symmetric (e.g. a circle or ring) then the projections from any direction are very similar and in this way their right position can be determined by finding the suitable geometric transformations between images, which can be done by a kind of registration [3]. Actually, two methods have been implemented in our system. Both correction methods can be separated into two sub-steps:

   a) An estimation of parameters of the correction transformation.
   b) The execution of the corrections transformation, obtaining a new, corrected projection sequence employed in further steps.

   The difference between the two methods lies in the choice of the transformation:

   a) The first method can be applied if the positioning errors are the same during each acquisition and they can be described such that the projection of the rotation axis translates only in the projection images (along

a sine curve). In this case the necessary transformation is the translation of each projection image. Its parameter can be estimated from the projection images of a previous experiment and the actual projections can be moved to the right positions.

b) The second method presumes that the projections of the object are very similar (e.g. the projections of a tube from directions perpendicular to the tube's axis) and by performing registration on all projections (see [13]) we can estimate the correct settings of the projections. In our case the transformation is rigid. If some of the projections are to be translated or rotated following the registration then our correction program is able to perform the necessary geometric transformation.

4. *Homogeneity correction.* Sometimes the detector plane is not uniformly sensitive in the whole field of view. This problem can be lessened if an 'empty' image is available. The empty image is acquired by imaging a homogeneous neutron flux. If the detector system is uniformly sensitive then this image is almost constant. Otherwise, the empty image shows how much correction (multiplication) is necessary, pixel by pixel on each projection, in order to obtain more constant images. The correction can be described mathematically in the following way. For each pixel i of all P_ϑ projections

$$R_\vartheta(i) = P_\vartheta(i) \cdot \frac{1}{P_{empty}(i)},$$

where R is the homogenity corrected image, and P_{empty} is the empty image.

5. *Intensity correction.* It might happen that the total intensities of the projection images vary during the acquisition period. The reason could be variations in the neutron flux or the electronics of the camera. When this occurs the images should be multiplied by suitable constants such that the average intensity in the background area is roughly the same. After this correction the flickering, which is often visible while playing the projection sequence like a movie, diminishes. This correction step can be divided into two sub-steps as well:

   a) The calculation of the correction factors for each projection on a selected background area.

   b) The execution of the correction method, which yields a new corrected projection sequence for further correction steps and reconstruction.

6. *Isolated points.* The neutron projections often contain white pixels (as shown in Fig. 11(b)) and some statistical noise, which both appear as isolated points having a very different intensity value compared with its neighborhood. In order to eliminate this kind of noise in the projections we performed thresholded median filtering. For each pixel i of each P_ϑ projection

$$R_\vartheta(i) = \begin{cases} P_\vartheta(i) & \text{if } |P_\vartheta(i) - \text{med}(\text{NRH}(P_\vartheta, i, n))| \leq \text{thr}, \\ \text{med}(\text{NRH}(P_\vartheta, i, n)) & \text{otherwise}, \end{cases}$$

where R is the corrected image, med(.) is the median operator, thr is a suitable threshold constant, and $\mathrm{NRH}(P, i, n)$ is a set which contains the intensity values of the n-neighborhood pixels of i in the image P. Such an n-neighborhood for $n = 8$ is depicted below.

The results of these pre-processing steps are given in Section 7.

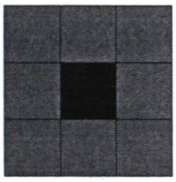


Figure 4: The 8-neighborhood of the black pixel is represented by gray pixels.

5 The reconstruction methods

Both methods essentially treat the reconstruction problem as an optimization task. Then we have to minimize the following objective functional

$$\Phi(f) = \sum_{\vartheta} \| [\mathcal{R}f](\vartheta) - P_\vartheta \|^2 + \gamma \|f - f_0\|^2, \tag{4}$$

where P_ϑ denotes the input projection of angle ϑ, f is the two-dimensional image function that approximates the solution, $[\mathcal{R}f](\vartheta)$ denotes the projection of the image f taken at angle ϑ, $\|.\|$ is the Euclidean norm, and $\gamma \geq 0$ is the so-called regularization parameter. Lastly f_0 is the prototype function. It is an image function like f that has the same domain and range, and is similar to the expected reconstruction result.

In (4) the first term is a function, which represents the distance between the projections of the current f and the given projection data P_ϑ. This term expresses how well the projections of the current f approximate the given projections. It should be noted that such reconstruction problems occur quite frequently; some typical applications are described in [5], say.

As mentioned earlier, the smaller the number of projections used, the more a priori information should be exploited. Some pieces of a priori information can be incorporated into the second term of (4). One important piece of information – if the prototype of the object is known – is a formula for f_0. The difference between the actual f and a given prototype object f_0 can be described in the second term. If the object f_0 is the zero image, then it tells us that such a solution is sought which is smooth enough. Finally, the non-negative γ regularization parameter is suitable for balancing the relative importance of the first and second terms. If γ is big, the optimization procedure returns a solution that is more similar to the prototype f_0 and less suitable for the given projection data.

Two reconstruction methods were implemented. The basic difference between them lies in their choice of object representation. The first one is a pixel-based method, while the second one is a kind of parameterized object reconstruction. Both methods reduce the reconstruction to the optimization problem described above, which is solved by applying simulated annealing (SA) [11]. Since simulated annealing is a statistical iterative optimization technique, the result is achieved through a sequence of approximating images, where the $(l+1)^{th}$ image is constructed by modifying the l^{th} image according to a predefined modification rule. The two reconstruction methods basically differ in the representation of the object f and choice of the modification rule used.

5.1 Pixel-based method

In this method the image f is represented as a *digital image*. The objects take their intensity values from a predefined known discrete set of attenuation coefficients $\{\mu_1, \mu_2, \ldots, \mu_d\}$, where d is the number of homogeneous materials from which the object is made. In the case of binary images the set of attenuation coefficients is $\{0, \infty\}$, thus the pixel values can be 0 or 1. In the SA procedure the pixel values will be modified. The modification rule used is simple. Let us take a pixel of f at random and change the 0 or 1 intensity to the other intensity value. More generally, if the image is not binary then the modification rule we apply changes the intensity value μ_i to μ_k at random, where $k \neq i$ and $1 \leq k \leq d$.

The annealing schedule (i.e. setting the parameters) of SA is a cornerstone of successful optimization. (Parameters needed in SA being described in detail in [12], say). One of the most important parameters is the initial temperature. If it is too high the optimization procedure can be too slow. However, with a low temperature there is a possibility that the algorithm will not find the global minimum but will get stuck in a local one. As was pointed out in [4], the optimal initial temperature is 4 degrees centigrade in practice. In this case the choice of the initial image can be arbitrary.

Another important parameter is the annealing function, which determines the speed of the temperature reduction. If the temperature decreases too quickly, the optimization can stop again in a local minimum, but if it decreases too slowly then slow annealing process decelerates the algorithm. The experimentally chosen cooling strategy in our system is defined as follows:

$$t(m+1) = t(m) \cdot c \ ,$$

where $t(m)$ returns the temperature (in degrees centigrade) in the mth iteration, and $c \in]0,1[$ is a feasible constant taking its value from the interval $[0.85, 0.95]$.

The optimization can be accelerated if the initial image is not far from the solution. Then the initial temperature can be decreased and SA finds the optimum faster than it does when starting it from an initial temperature of 4 degrees centigrade. What the right initial temperature is should be investigated further, however.

5.2 Parameter-based reconstruction method

In this method the image function f represents a three-dimensional object consisting of geometric primitives (cylinders and spheres) determined by a small number of parameters. More precisely, we assume that the object to be reconstructed consists of a tube encompassing a solid cylinder called the interior (i.e. the inner space of the tube), which contains a known number of disjoint solid spheres or cylinders made of homogeneous materials. In the present implementation the range of f may consist of at most four distinct values. The object to be reconstructed should consist of the following parts:

- The tube;
- the interior encompassed by the tube;
- the spheres and cylinders contained in the interior;
- and the background surrounding the object, which is usually air or a vacuum.

It should be noted that our aim was to perform a so-called *truly 3D reconstruction*, as opposed to our earlier approaches [9, 10] when the 3D problem was reduced to 2D sub-problems, that is to the reconstruction of 2D cross-sectional slices of the object.

In addition to the a priori information mentioned above, other assumptions were made here. These are the following:

- The cylinders of the wall of the tube are concentric.
- The interior of the tube may contain either spheres or cylinders, but not both. Moreover, their number is known.
- The absorption coefficients are known, at least approximately. In addition, the absorption coefficient of the background is zero or nearly zero, making its effects negligable.
- The projections are taken along parallel beams using equidistant detector spacing.
- The axis of rotation of every cylindrical component is perpendicular to the projecting beams.

Every sphere in the object is uniquely determined by its radius and the coordinates of its center, which are the parameters of the sphere. Cylinders and the tube can be parameterized in a similar way. Furthermore, the absorption coefficients (μ_1, μ_2, μ_3) corresponding to the three materials (i.e. the material of the tube, that of the interior, and that of the spheres and cylinders) are also considered as parameters. (Note here that the absorption of the background material is assumed to be zero.) Such objects can then be uniquely described by a vector of parameters called a *configuration*.

Our second reconstruction method is also an iterative procedure. We start with an initial configuration and during the iterations the current configuration (i.e. the current image) is altered in order to get a hopefully better approximation of the object to be reconstructed. The current image function f is considered *admissible* if certain geometric constraints are met. These are the following:

- The cylinders of the tube remain concentric, the tube is contained by image f, and it has a non-negative wall thickness. Note that allowing a zero thickness may be useful in the case when the object is in fact a solid cylinder containing some spheres or cylinders. This property is used, for instance, in the physical experiments shown in Section 9.2.
- The radius of the interior equals the inner radius of the tube, so that the interior completely fills the inner volume of the tube.
- All distance parameters (namely radii, heights) are positive numbers.
- All the spheres and cylinders are located within the interior, and they are pairwise disjoint.

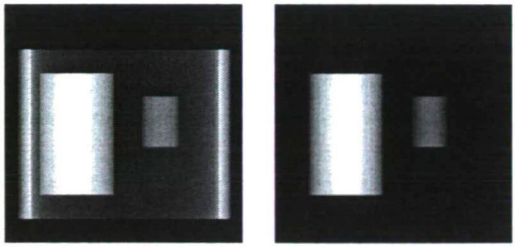
It is a relatively straightforward task to check all of these conditions during the modification of the current configuration.

It is often desirable to generate random configurations automatically for testing and simulation studies – see Section 9.1, say. Our implementation allows this by letting the user specify how the various parameters may vary, then randomly generating a new configuration based on these settings, while maintaining the above-mentioned geometric restrictions.

The optimization of the functional $\Phi(f)$ is performed in the parameter space. At each iteration step, a new configuration is built by modifying one of the parameters of the current configuration. This is done so that every parameter may be selected with the same probability (i.e. uniformly), and the amount of modification is chosen randomly (again, uniformly) from the whole set of feasible values. Those configurations which are inadmissible are always rejected. The projections $[\mathcal{R}f](s,\vartheta)$ of f are then calculated analytically. It should be remarked that the computation of $\Phi(f)$ can be speeded up by some orders of magnitude by restricting calculations to the sub-domain in which the value of $\Phi(f)$ has changed in the last iteration step.

In order to keep the number of configuration changes as low as possible it is important to find a good initial configuration. The SA algorithm can probably find the optimum in a fewer number of steps when an initial image is quite near the optimum. The initial configuration for our system is constructed as follows.

1. First, the initial locations of the tube and its interior are estimated from the projections (see Fig. 5(a)). This is done after using filtered versions of the input projections, obtained by applying a Gaussian and several averaging filtering kernels to every projection. The boundary of the projection of the tube, namely the coordinates of its centerline, its radius, and its upper and

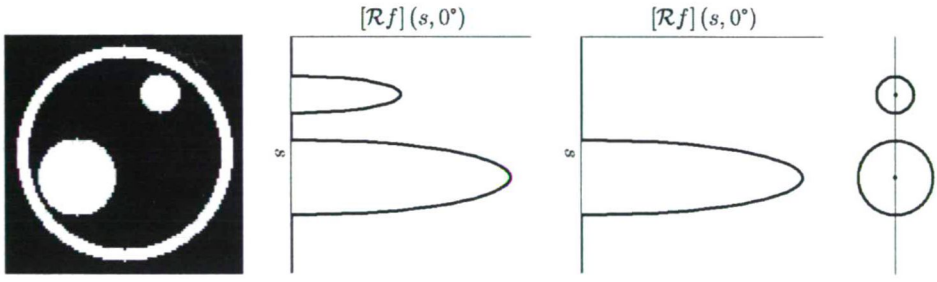


(a) (b)

Figure 5: (a) One of the projections of the original object. (b) One of the reduced projections generated by subtracting the projection of the tube and its interior from the projection shown in (a).

lower ends can be found by defining a threshold for the background noise, and scanning the filtered projections for the leftmost, rightmost, uppermost, and lowermost values above this threshold. The boundary of the interior of the tube can be determined in a similar way. The threshold is calculated as a user-specified percentage of the maximum value of the projections, based on an assumed level of noise.

2. Using the location of the tube and its interior, their projections can be subtracted from the input projections. This is shown in Fig. 5(b).



(a) (b) (c) (d)

Figure 6: (a) Cross-section of the original object depicted in Fig. 5(a). (b) Cross-section of the reduced projection shown in Fig. 5(b). (c) Elimination of the projection of a disc from the projection displayed in (b). (d) Discs detected in projection (b).

3. Next, the initial positions of the spheres or cylinders in the interior are to be guessed based on the reduced projections (see Fig. 6). In the present implementation this task has, for simplicity, been reduced to 2D sub-problems, but it may be possible in future to extend the process to 3D. Currently this procedure is performed in two steps:

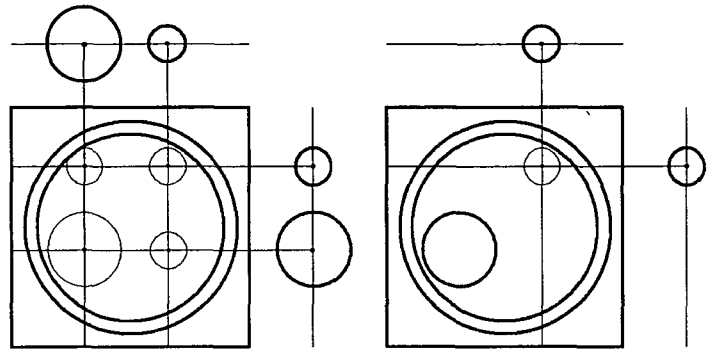


Figure 7: (a) Initial intersections of the cross-section shown in Fig. 5(a) using a horizontal and a vertical projection. (b) Deletion of intersections after choosing the disc with the largest radius.

a) First, the locations of discs are found in each cross-section separately (see Fig. 6(a)). That is, the intersections of the spheres or cylinders with the 2D cross-sections orthogonal to the axis of rotation have to be found. This is accomplished by the following greedy algorithm based on geometric considerations:

   i. Possible projections of discs are located within the 1D projections of the cross-sections (see Fig. 6(b)) by the application of model-fitting, i.e. disc parameters are estimated. In particular, a deterministic relaxation algorithm called iterated conditional modes (ICM, [2]) is used to find the parameters of the projection of a disc which best fits the 1D reduced projections. After storing these parameters, the projection of the disc is eliminated from the already reduced projections (see Fig. 6(c)). This process is repeated until no more discs can be detected (see Fig. 6(d)).

   ii. The centers of the discs found in the previous step are 'projected back' into the plane of the 2D cross-section, thus forming several pairs of intersection points and radii of the discs associated with them (see Fig. 7(a)). Those intersections whose distance from one another is below a given limit will be merged into a single intersection. Whatever the case, the radius associated with a particular intersection is calculated by taking the minimum of the radii of the corresponding discs found in the 1D projections.

   iii. The discs actually located in the 2D cross-section are selected by a greedy strategy: The center of a candidate disc is chosen to be the intersection which was defined by the maximum number of possible projections. Should there be more than one such intersections, the one with the largest radius associated with it will be taken. The in-

tersection chosen is removed instantly from the list of intersections. We should remark here that some additional intersections may be deleted as well in order to retain consistency between the intersections and the 1D projections (see Fig. 7(b)). If this disc results in an admissible configuration, it will be added to the 2D configuration of the cross-section. Otherwise, another intersection will be chosen. This procedure is repeated until no more intersections are left.

b) Lastly, the initial positions of the spheres or cylinders are determined from the discs found in neighboring cross-sections. Specifically, the discs detected in the previous step are treated as solid cylinders of unit height. An auxiliary 3D configuration can be built by stacking the cross-sections onto each other, that is placing these discs into a 3D coordinate system. This configuration is then examined to find candidate spheres and cylinders that will be included in the initial configuration.

6 Modelling the noise

In order for the conditions to be as realistic as possible in the simulation experiments, artificial noise was generated with a uniform distribution and then added to the projections. An example of this is shown in Fig. 8.

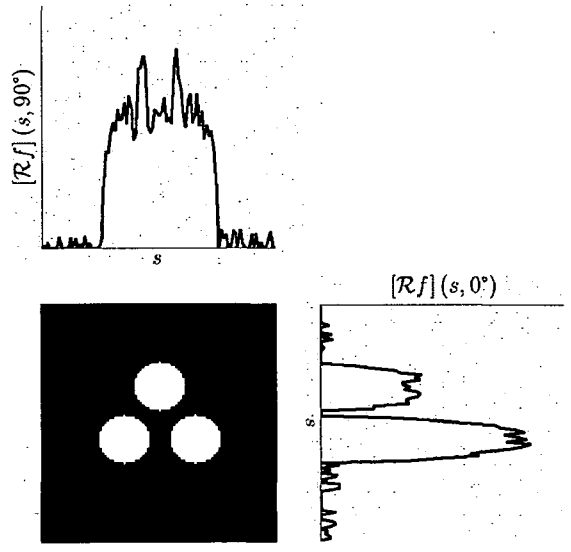


Figure 8: Horizontal and vertical projections of a binary image with 10% noise.

7 Results of correction methods

The pre-processing steps discussed in Section 4 were tried out on the projections of a VIDICON tube (see Fig. 9). There were 360 projections taken in 1° angular steps using X-ray radiation.

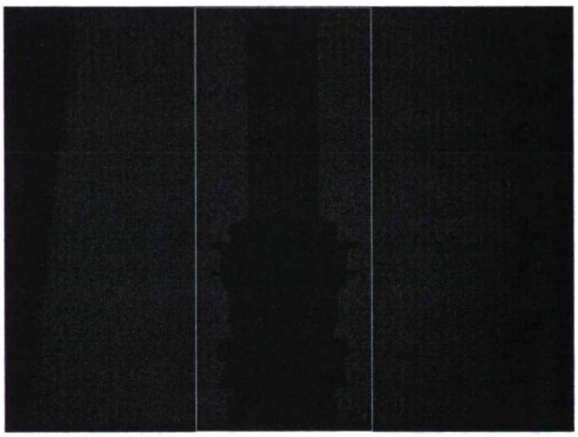


Figure 9: One of the projections of the VIDICON tube (height $\approx$ 10 cm, width $\approx$ 4 cm). The white lines indicate the cropped area.

The first step involved cropping the relevant area from the original projections when the object to be reconstructed covered only a small portion of the projections.

The second step of pre-processing was the correction of the motions during the image acquisition. As a result of the analysis of the projections (using registration) we found that in this case each projection had to be translated along a sine curve having a phase of 90° and amplitude of 2 pixels.

When an empty projection image is available as well (as was the case here), homogeneity correction can be performed. Homogeneity correction eliminates the effects of the non-uniformly sensitive detector system.

The results of the pre-processing steps (cropping, motion and uniformity corrections) can be seen in Fig. 11.

The intensity correction step of the pre-processing multiplies the projections by suitable coefficients such that the total intensity of the corrected projections is practically constant for each projection (see Fig. 10).

Finally, the isolated points correction was applied to each projection. This correction means the application of a thresholded median filter (8-neighborhood, thr = 20 assuming a maximum intensity level of 255). In Fig. 11(c) we see that a part of the noise was eliminated from the projections and the reconstructed cross-section became a bit more homogeneous compared with Fig. 11(f). This homogeneity feature is especially noticeable in the interior of the dashed circle.

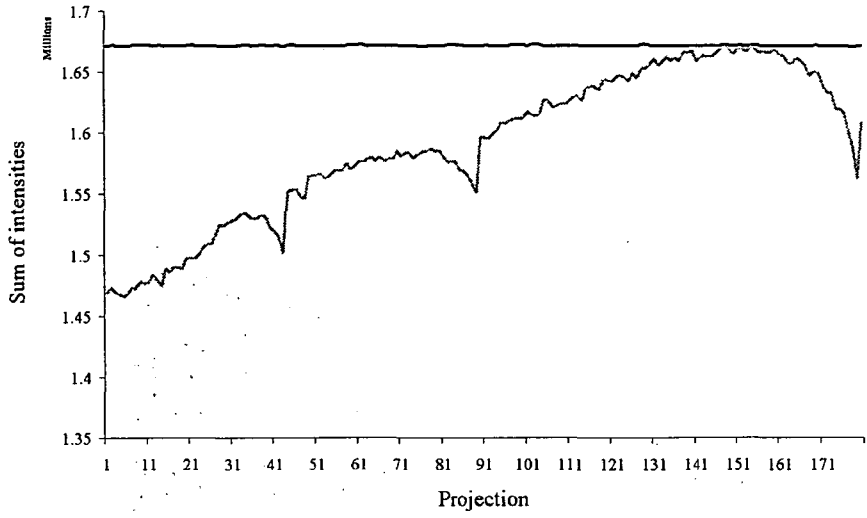


Figure 10: Total intensity values in the first 180 projections before (gray line) and after (black line) intensity correction.

8 Results of the pixel-based reconstruction method

In this section we demonstrate the efficiency of the pixel-based reconstruction method. First, software simulated phantom studies will be presented where we show the effects of changing different parameters on the reconstruction. Then some preliminary results on some neutron tomography data are described in Section 8.2.

In order to measure the accuracy (precision) of the reconstruction quantitatively, we used the relative mean error (RME for short), which is defined here as

$$\mathrm{RME} = \frac{\sum_i |f_i^{\mathrm{o}} - f_i^{\mathrm{r}}|}{\sum_i f_i^{\mathrm{o}}} \cdot 100\% \,,$$

where f_i^{o} and f_i^{r} denote the value of the ith pixel of the original and the reconstructed image functions, respectively. It is quite clear that RME ≥ 0 and a smaller RME value means a better reconstruction result.

8.1 Simulation studies

The simplest case is when the range of the image function to be reconstructed contains only two values, 0 and 1. The results of such reconstructions can be seen in Figs. 12 and 13.

First, we investigated how the number of projections influences the reconstruction (see Fig. 12). It is conspicuous that the pixel-based method was able to almost

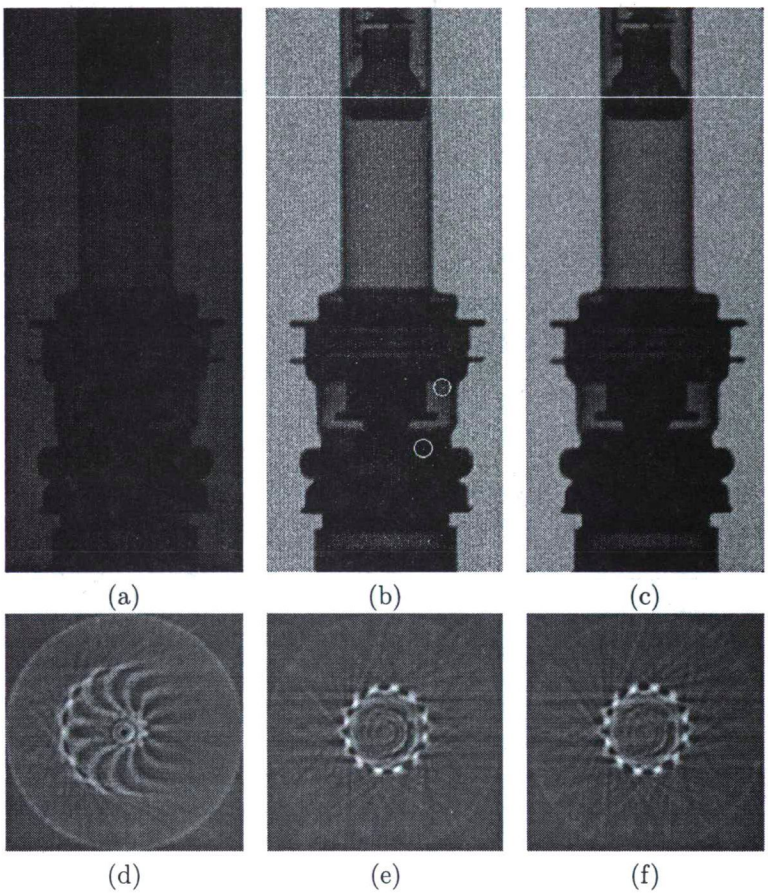


Figure 11: (a) One of the projections of a VIDICON tube after cropping having the size 241 × 572. (White line indicates the cross-section shown below.) (b) The same projection after motion and intensity corrections (the white circles indicate isolated points to be corrected in the next step). (c) The same projection an in (b) after homogeneity and isolated points corrections. (d) Reconstruction (241 × 241) of the cross-section shown after cropping. (e) Reconstruction of the cross-section shown after motion and intensity corrections. (f) Reconstruction of the cross-section shown after homogeneity and isolated points corrections. The reconstruction was performed by the software package SNARK93 [15] (filtered backprojection, cosine filter, cut-off frequency 0.5, Lagrange interpolation).

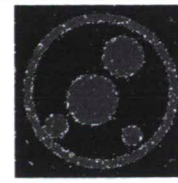
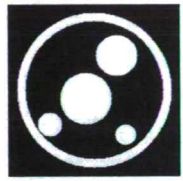
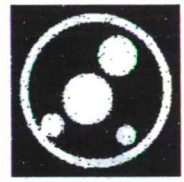
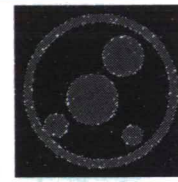
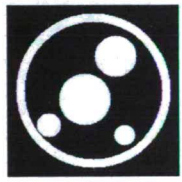
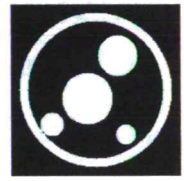
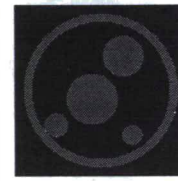
# of projs	Original	Result	Difference	RME (%)
8				12.57
10				6.98
12				0.10

Figure 12: Reconstruction of circles from different numbers of projections by the pixel-based method (0% noise, 200 measurements/projection). First column: number of projections. Second column: original object. Third column: reconstructed images. Fourth column: RME of the original and reconstructed images (black/gray: when the corresponding pixels are black/white in both the original and reconstructed images; white: when the corresponding pixel intensities are different). Fifth column: RME of the original and reconstructed images.

reproduce the original phantom with just 12 projections.

We also studied the effect of noise on the reconstruction. 0%, 10%, and 40% noise was added to the exact projections and then the reconstruction was done with the noisy projections as well (see Fig. 13). As expected, the quality of the reconstruction grew worse the higher the level of noise in the projections was chosen. The same can be seen from the RMEs. But the object, however, is still recognizable even with 40% noise.

8.2 Physical experiments

We had the chance to test our reconstruction methods with real physical data as well. One of them was the battery from a pacemaker (Fig. 14(a)). As can be seen in the image, especially in Fig. 14(b), the projections were almost noiseless and of a good quality. In order to compare the results of the classic FBP technique with our pixel-based one, we reconstructed several slices. One of them is shown in Fig.

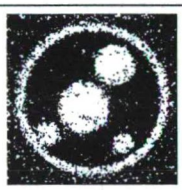
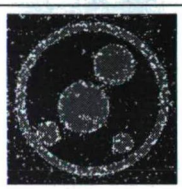
Noise (%)	Original	Result	Difference	RME (%)
0				0.12
10				20.64
40				34.63

Figure 13: Reconstruction of circles from exact and noisy projections by the pixel-based method (16 projections, 200 measurements/projection). First column: noise level. Second column: original object. Third column: reconstructed images. Fourth column: difference images. Fifth column: RME of the original and reconstructed images.

14(a), and the corresponding reconstruction result, based on 200 projections and using the FBP method of SNARK93 [15], can be seen in Fig. 15. This result can be considered quite favourable as the object is reasonably distinguishable.

In order to compare the results of the classical method we perfomed an FBP using 20 projections, and we ran our pixel-based technique with the same input data as well. In Fig. 14(c) the results of FBP are presented, which contain streaks due to the small number of projections used. In the case of the pixel-based method (Fig. 14(d)) these streaks are absent, but the quality of the results are still worse than the previous one. The reason might be that the DT algorithm did the reconstruction using only 21 intensity levels, but the object could not be considered one which satifies our basic assumption: the object consists of only a few *inhomogeneous* materials. This inhomogeneity or irregular material distribution is quite visible in Fig. 15. In addition, we were not aware of the exact absorption coefficients. If the object had been made of homogeneous materials, we would have hoped for better results.

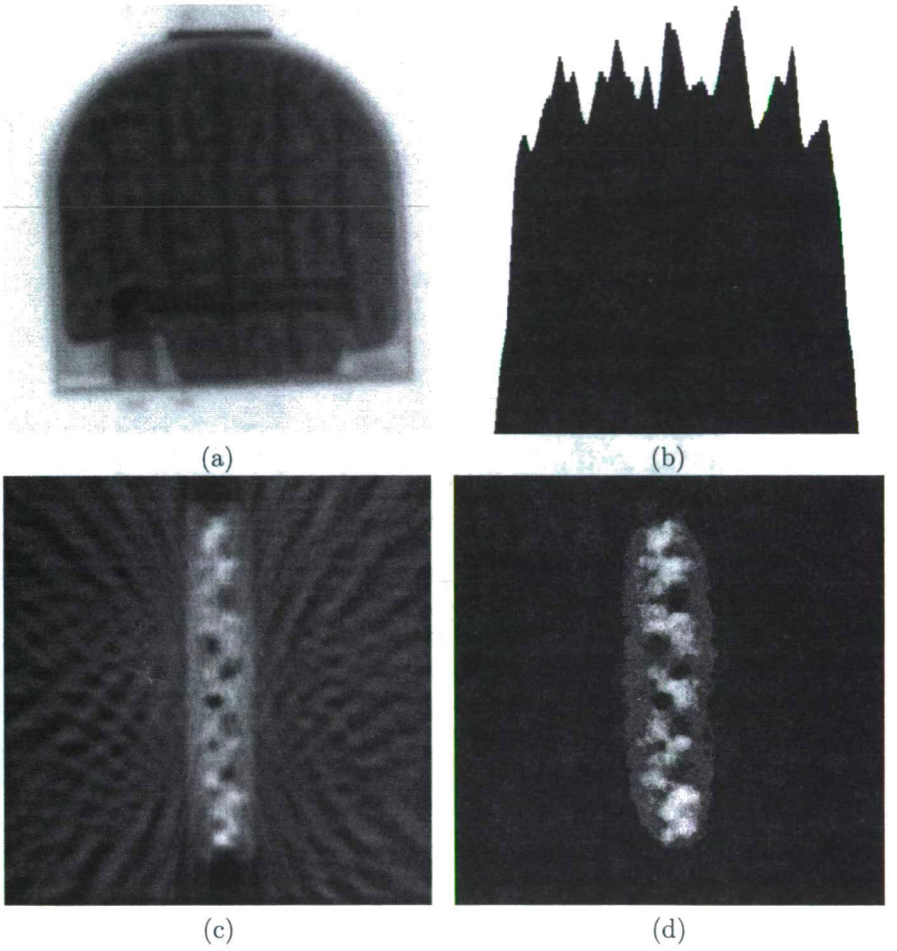


Figure 14: (a) One of the neutron projections of a pacemaker battery. (b) The bar diagram representing the intensity values of the projections in the indicated row in (a). (c) Reconstructed cross-section in the position indicated in the row in (a) (FBP method 20 projections, cosine filter, cutoff 0.5, Lagrange interpolation as done in SNARK93 [15]). (d) Image of the same cross-section reconstructed via the pixel-based method using 20 projections and 21 intensity levels.

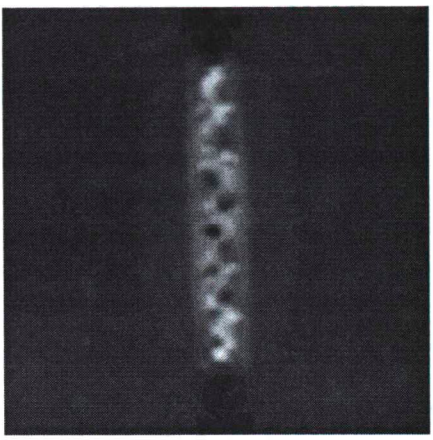


Figure 15: FBP reconstruction (as done in SNARK93 [15]) of the cross-section shown in Fig. 14(a) using 200 projections.

From the point of view of timing the technique cannot be said to be fast. Its speed greatly depends on the input data. In accordance with our experiences the technique terminates within 60 seconds in case of noiseless phantom images on a 3 GHz Intel Pentium 4, where the size of the images is at most 400×400 pixels using less than 20 projections. However, a reconstruction using real projections can cost up to 5–10 minutes.

9 Results of the parameter-based reconstruction method

In this section we will present the results of the parameter-based reconstruction method. First, simulation studies will be discussed, focussing on the effects of various parameters on the reconstruction. These parameters include the geometric complexity of the object, the amount of noise in the projections, and the number of projections. Afterwards, some results of the physical experiments will be given.

In order to assess the accuracy of this method quantitatively, the RME measure defined in Section 8 was used throughout the experiments with a slight modification: the image function was converted to a $\{0, 1\}$ binary image before calculations. It should be mentioned that a more precise way of measuring would be to compute RME analytically instead of using the digitized image, which is one of our future plans. The models of objects are visualized using the Virtual Reality Modeling Language (VRML97 [16]).

9.1 Simulation experiments

Software experiments were performed in order to investigate the effects of key reconstruction parameters: the geometric complexity of the object to be reconstructed, noise level, and the number of projections. Another important thing we need to know is how the value of the objective function $\Phi(f)$ changes during iterations. A typical plot of $\Phi(f)$ is shown in Fig. 16, produced during the reconstrucion of 3 spheres from 4 noisy projections. As expected, the objective function decreases rapidly at the beginning of the optimization, and more slowly near the global minimum.

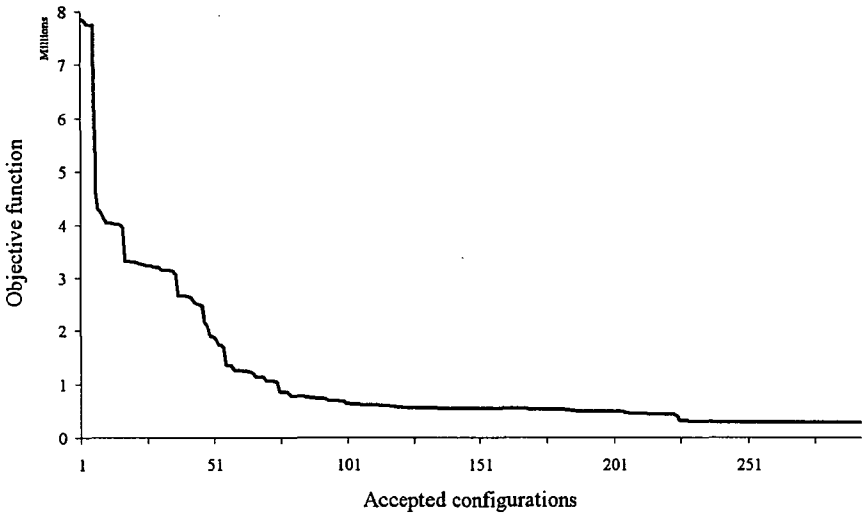


Figure 16: The value of the objective function $\Phi(f)$ as a function of the number of accepted configurations. (The total number of accepted configurations was 291. The reconstruction of 3 spheres using 4 projections degraded with 10% noise, with 100×100 measurements/projection.)

The aim of the first experiment was to study the influence of the geometric complexity of the object (see Fig. 17) using only 2 projections degraded with 10% noise. As can be seen in Fig. 18, it is hard to get an acceptable result with just 5 spheres. The algorithm, however, can successfully reconstruct fewer spheres with good precision.

We also looked at the effects of noise on our reconstruction using the parametric object shown in Fig. 19. The noise level was again set to 0%, 10%, or 40% as we did in the simulation experiments using the pixel-based reconstruction method. It is remarkable that the parameter-based reconstruction method still works even with 40% noise (see Fig. 20).

In the last simulation experiment we analyzed the impact of the number of projections on the results (see Fig. 21). Since real physical measurements are usually

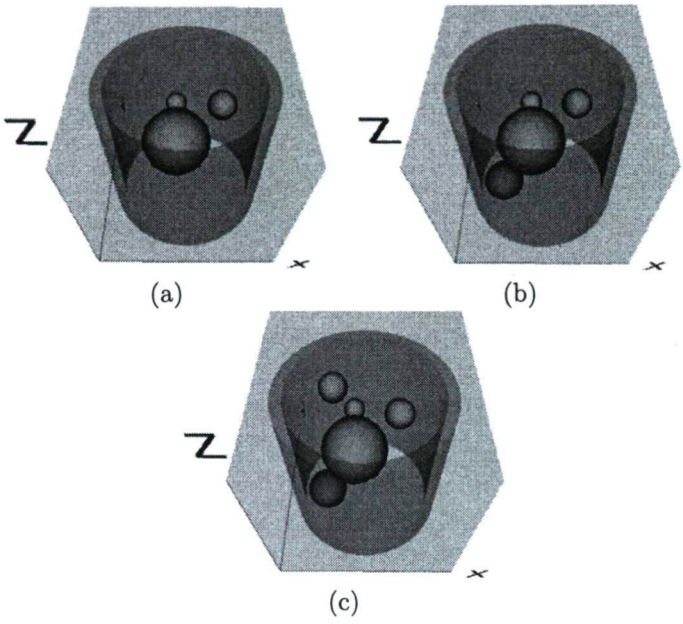


Figure 17: Perspective view of 3D parametric phantoms consisting of (a) 3, (b) 4, and (c) 5 spheres in a tube.

distorted by some noise, the same experiment was performed using noiseless projections as well as adding 10% or 40% noise to the projections. It is quite apparent here that the reconstruction of these parameterized objects does not depend much on the number of projections used. As Fig. 21 indicates, it is usually sufficient to use just two projections for simpler objects (e.g. three spheres). Actually, having more projections does not necessarily improve the precision of the result by much. At the same time, the algorithm is more sensitive to the geometrical complexity of the object to be reconstructed, as is demonstrated by the third row in Fig. 18.

9.2 Physical experiments

In addition to the software-generated data described above, the effectiveness of the parameter-based reconstruction method was also evaluated using physically measured data. However, before proceeding with a discussion of the results, two remarks should be made. First, due to the limitations of the imaging system and the measurement errors, the projections were distorted and quite noisy. To help the algorithm produce the best results possible, it was necessary to perform the pre-processing steps described in Section 4. Second, as the exact values of the absorption coefficients were unknown, they had to be estimated here.

The first physical experiment was performed on a phantom object called the reference cylinder, a diagram of it being shown in Fig. 22. The object is a solid

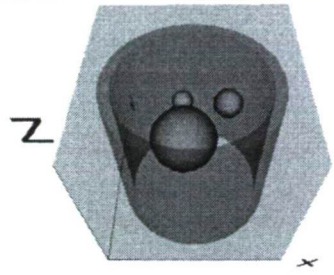
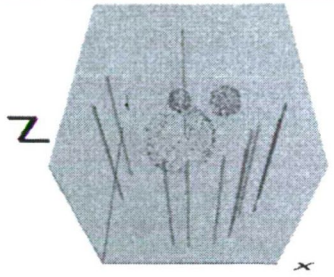
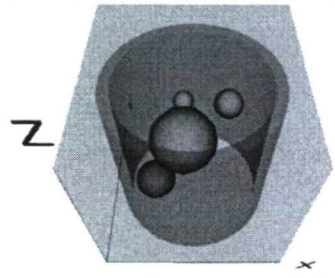
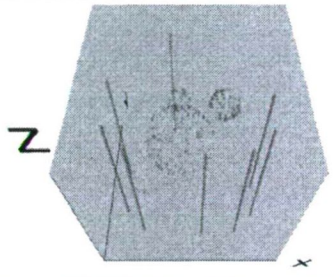
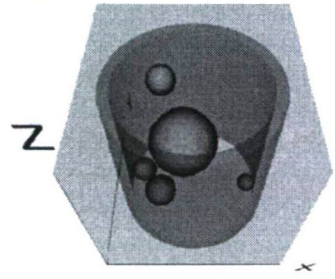
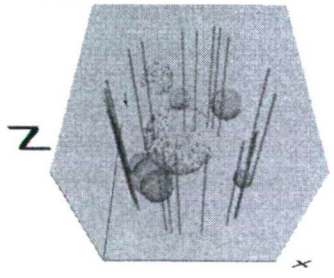
# of sph.	Result	Difference	RME (%)
3			0.97
4			0.66
5			6.27

Figure 18: Reconstruction by the parameter-based method using a different number of spheres (original object: see Fig. 17; parameters: 10% noise, 2 projections, 100×100 measurements/projection). First column: number of spheres. Second column: reconstructed object. Third column: difference between the reconstructed and original object (only mismatching voxels are painted). Fourth column: RME of the original and reconstructed image.

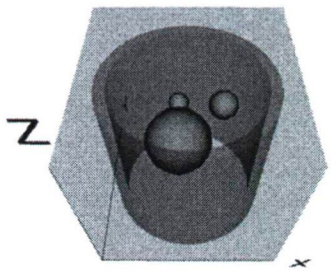


Figure 19: Perspective view of the 3D parametric phantom object consisting of 3 spheres in a tube.

cylinder made of Plexiglas, and it contains three holes of different diameter and various depths in an asymmetric arrangement. The lower part of the hole with the largest depth was filled with aluminum screws. This region is clearly visible on the projection images, which were taken using an X-ray source. One of these projections is shown in Fig. 23(a) after applying every pre-processing step except that for the filtering of isolated points.

Since our model assumes that the cylindrical holes are filled with the same material, the lower half of the projections had to be thrown away (see Fig. 23(b)). The size of the projection images was 155×212 pixels before and 155×113 pixels after cropping, respectively. It is evident that the remaining part of the object is suitable for discrete tomographic reconstruction, since it consists of two homogeneous regions containing air and Plexiglas (background and holes, respectively).

The model reconstructed using 4 projections is illustrated in Fig. 23(c) to Fig. 23(e). Since the exact structure of the object was known, it was possible to create an 'original' model, and thus to measure the precision of the reconstruction. The difference between the original and the reconstructed model is shown in Fig. 23(f), with an RME of 2.45%. This relatively high value is due to two facts: the projections were fairly small and noisy, and the exact value of the absorption coefficient of Plexiglas was unknown.

A phantom object very similar to the one mentioned above was used in the second physical experiment. The structure of this object was identical to that shown in Fig. 22, but the solid cylinder here was made of aluminum. The hole with the biggest diameter was partly filled with acetone and the two others contained water.

At first sight this object seems unsuitable for our parametric object model, since the bores consist of three homogeneous regions: acetone, water, and air. It turned out, however, that the absorption coefficients of the acetone and of the water are almost equal for the neutron rays used for image acquisition. The size of the projection images was 365×400 pixels, and one of them can be seen in Fig. 24(a) after applying the same pre-processing steps as in the first experiment. Moreover, those sections of the bores which contained air are hardly recognizable. Hence the object can be considered to be composed of three approximately homogeneous materials: aluminum, fluid, and air (background).

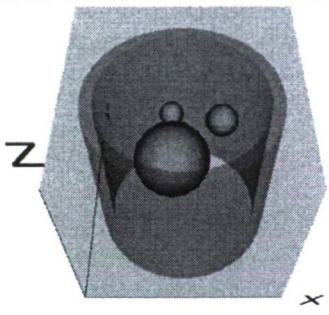
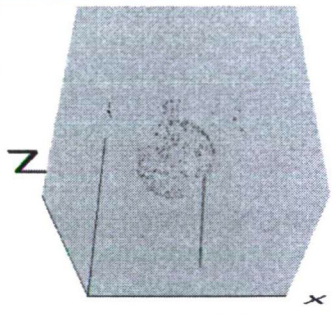
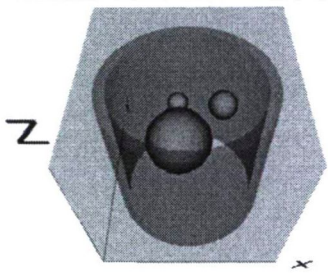
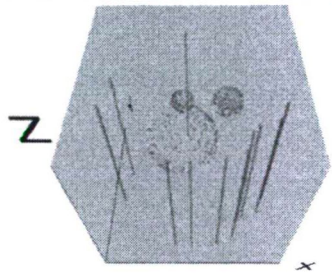
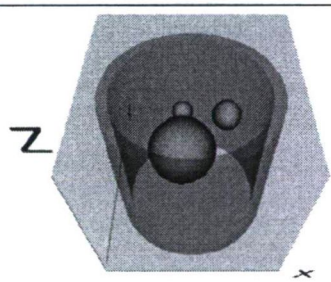
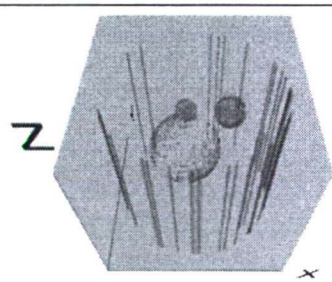
Noise (%)	Result	Difference	RME (%)
0			0.22
10			0.97
40			2.94

Figure 20: Reconstruction by parameter-based method from noise free and noisy projections (original object: see Fig. 19; parameters: 2 projections and 100×100 measurements/projection). First column: noise level. Second column: reconstructed object. Third column: difference between the reconstructed and original object. Fourth column: RME of the original and reconstructed image.

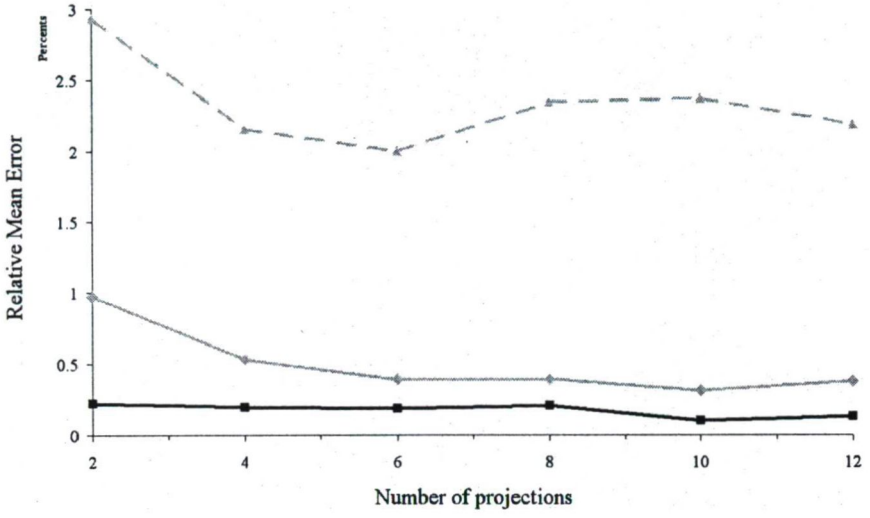


Figure 21: The RME value versus the number of projections without noise (black curve), with 10% (solid gray curve) and with 40% noise (dashed gray curve), respectively. (Reconstruction of 3 spheres using 2, 4, 6, 8, 10 and 12 projections, and 100×100 measurements/projection.)

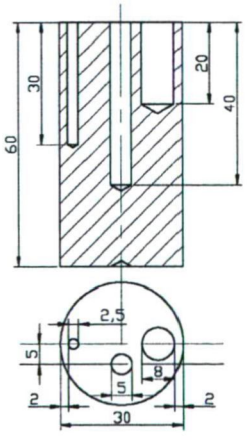


Figure 22: Diagram of the phantom object used in the experiments (dimensions are shown in mm.)

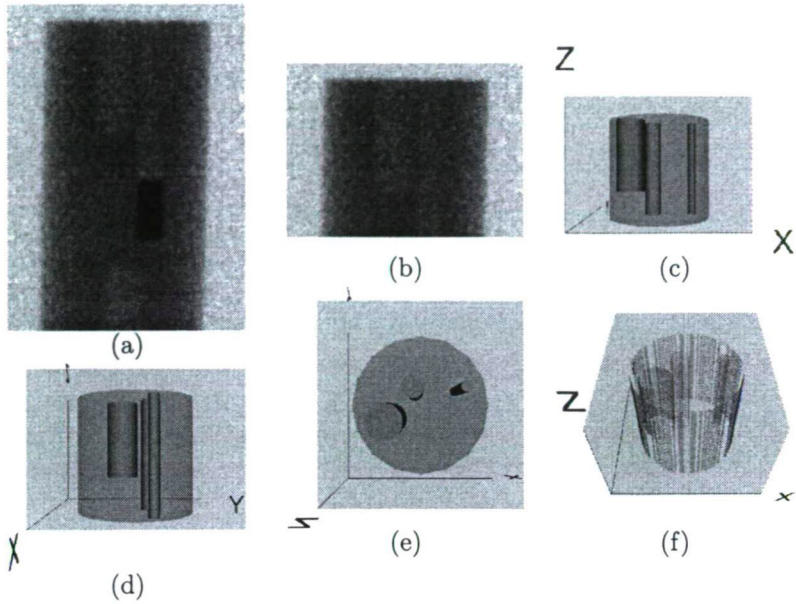


Figure 23: Projection and reconstruction results of the Plexiglas object given in Fig. 22. (a) One of the original projection images. (b) One of the cropped projection images. (c) 0° view of the reconstructed model. (d) 90° view of the reconstructed model. (e) Top-down view of the reconstructed model. (f) Difference between the reconstructed model and the original one (RME = 2.45%).

The model reconstructed using 4 projections is shown in Fig. 24(b), (c) and (d). As the structure of the object is identical to the one used in the previous experiment, an 'original' model can be created again, thus allowing one to test the precision of the reconstruction. The difference between the original and the reconstructed model is shown in Fig. 24(e). Even though the precise radius of the bore with the smallest diameter in the reconstructed result is somewhat larger than the real one, the RME is only 1.01%. This is because the RME is calculated using the digitized image function, and the number of voxels is much larger than in the former case. The discrepancies between the original and the resulting models are due to the following: the axis of rotation was precessing, the exact values of the absorption coefficients were unknown and, despite our earlier assumption, the absorption coefficients of water and of acetone were quite different.

It is notable that the parameter-based method performs somewhat faster than its pixel-based counterpart. In particular, typical running times are a few (5–10) seconds on a 3 GHz Intel Pentium 4, using 4 projections and assuming that the size of the image function to be reconstructed is at most $100 \times 100 \times 100$ voxels. On the other hand, when the image function is as large as that in the last physical experiment (i.e. $365 \times 365 \times 400$ voxels), it may take even 3–5 minutes to complete the reconstruction process.

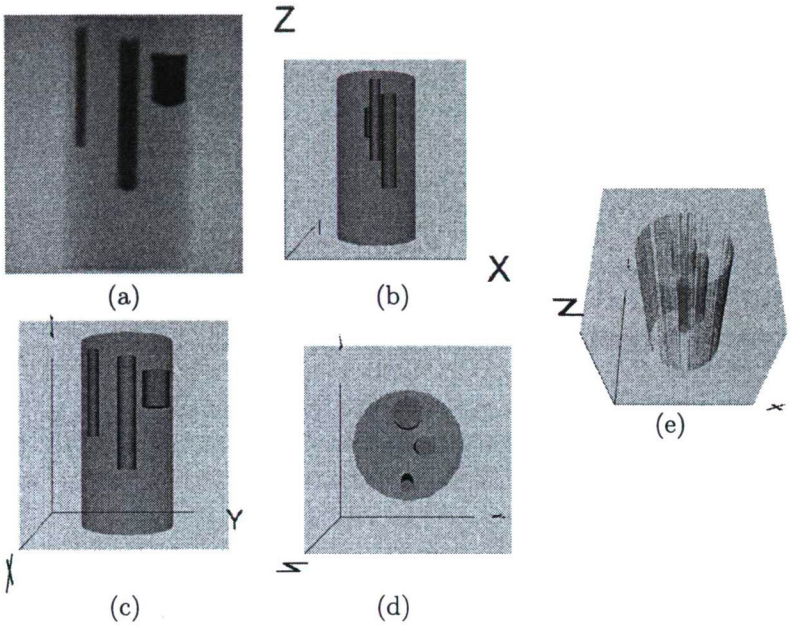


Figure 24: Projection and reconstruction results of the aluminum object given in Fig. 22. (a) One of the original projection images. (b) 0° view of the reconstructed model. (c) 90° view of the reconstructed model. (d) Top-down view of the reconstructed model. (e) Difference between the reconstructed model and the original one (RME = 1.01%).

10 Future plans

Though the techniques presented here produced promising and even acceptable results, we are planning to improve several things in the future. It would be desirable to make pre-processing as automated as possible in both methods, and for the absorption coefficients to be estimated automatically – for instance by using some kind of calibration procedure. For the pixel-based method, the most useful improvements would be to make it less sensitive to noise and to reduce the number of projections necessary for reconstruction. We would also like to make this approach work better for objects containing three or more materials. Further research could be done for situations where the projections are not a parallel beam but fan-beam, say. Finally, some practical extensions of the parameter-based algorithm might allow five or more materials, using more complex prior knowledge of the object, computing the RME analytically, and employing another kind of parametric object representation like a deformable model.

Acknowledgements

This work was supported by the NSF grant DMS0306215 (Aspects of Discrete Tomography) and the OTKA grant T 048476 (New Aspects and Applications of Discrete Tomography in Neutron Radiography). The authors wish to thank Márton Balaskó (KFKI Atomic Energy Research Institute, Budapest) and Prof. W. Treimer (Hahn Meitner Institut, Berlin) for providing us with the neutron and X-ray images of the objects used in the experiments. We would also like to thank David Curley for checking this paper from a linguistic point of view.

References

[1] M. Balaskó, A. Kuba, A. Nagy, Z. Kiss, L. Rodek, and L. Ruskó, *Neutron-, gamma- and X-ray three-dimensional computed tomography at the Budapest research reactor site*, Nucl. Instrum. Methods Phys. Res. A **542A** (2005), 22–27.

[2] J. Besag, *On the statistical analysis of dirty pictures*, J. Roy. Statist. Soc. Ser. B **48** (1986), 259–302.

[3] L. G. Brown, *A survey of image registration techniques*, ACM Computing Surveys **24** (1992), 325–376.

[4] S. Geman and D. Geman, *Stochastic Relaxation, Gibbs distributions, and the Bayesian Restoration of Images*, IEEE Trans. Pattern Anal. Machine Intell. **6** (1984), 721–741.

[5] G. T. Herman and A. Kuba (Eds.), *Discrete Tomography: Foundations, Algorithms, and Applications*, Birkhäuser, Boston, MA (1999).

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science **220** (1983), 671–680.

[7] Z. Kiss, L. Rodek, A. Nagy, A. Kuba, and M. Balaskó, *Reconstruction of pixel-based and geometric objects by discrete tomography. Simulation and physical experiments*, Electronic Notes in Discrete Mathematics **20** (2005), 475–491.

[8] A. Kuba, L. Rodek, Z. Kiss, L. Ruskó, A. Nagy, and M. Balaskó, *Discrete tomography in neutron radiography*, Nucl. Instrum. Methods Phys. Res. A **542A** (2005), 376–382.

[9] A. Kuba, L. Ruskó, L. Rodek, and Z. Kiss, *Application of Discrete Tomography in Neutron Imaging*, Proc. of 7th World Conference on Neutron Radiography, Rome, Italy (2002), 361–371.

[10] A. Kuba, L. Ruskó, L. Rodek, and Z. Kiss, *Preliminary studies of discrete tomography in neutron imaging*, IEEE Trans. Nucl. Sci. **52** (2005), 380–385.

[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys. **21** (1953), 1087–1092.

[12] N. Robert, F. Peyrin, and M. J. Yaffe, *Binary Vascular Reconstruction from a Limited Number of Cone Beam Projections*, Med. Phys. **21** (1994), 1839–1850.

[13] A. Tanács and A. Kuba, *Evaluation of a Fully Automatic Medical Image Registration Algorithm Based on Mutual Information*, Acta Cybernet. **16** (2003), 327–336.

[14] `http://www.inf.u-szeged.hu/~direct/`, Homepage of DIRECT framework, *A toolkit for testing and comparing 2D/3D reconstruction methods of discrete tomography.*

[15] `http://www.cs.gc.cuny.edu/~gherman/snark2001.html`, Homepage of the SNARK93 software system, *A Programming System for 2D Image Reconstruction from Projections.*

[16] `http://www.web3d.org/x3d/vrml/`, Homepage of VRML97, *Virtual Reality Modeling Language.*

Received July, 2005

Acta Cybernetica **17** (2006) 589–603.

MedEdit: A Computer Assisted Image Processing and Navigation System for Orthopedic Trauma Surgery*

Krisztián Ollé† Balázs Erdőhelyi†
Attila Kuba† Csongor Halmai† and Endre Varga‡

Abstract

The surgery of fractured bones is often a very complex problem. That is the reason why it would be beneficial to create a geometric and mechanic model of the bones before surgical intervention. The model geometry is based on the CT images of the patient and the known physical properties of the bone. A computerised system is presented here, called MedEdit, which helps a surgeon plan an operation. The system includes a Finite Element Analysis (FEA) program to measure the stress effects of the possible surgical solutions. Following the simulation and analysis of the behaviour of the modelled bone, surgeons can find the best surgical solution for the patient.

1 Introduction

The surgical repair of fractured bones is often a difficult task, and the fixation of these bones has to be planned very carefully. This is why trauma surgeons may use Interactive Image-Guided Surgery (IIGS) or Computer Aided Surgery (CAS) systems to improve surgical accuracy. By using these systems treatment can have several advantages. For example one can have a less invasive surgical approach, have less time consuming interventions due to better planning, and there can be a reduction in radiation exposure.

CAS technology was first introduced in neurosurgery. Here the neurosurgeons focused on pre-operative planning and an increase in intra-operative accuracy. By applying external frames to the patient's skull there is a possibility of locating targets within the brain to a high precision. This technique is called stereotactic neurosurgery [7]. The first application of CAS in orthopedic and trauma surgery

*This work was supported by the Grant OTKA T37840

†Department of Image Processing and Computer Graphics, University of Szeged, H-6720 Szeged Árpád tér 2. E-mail: {ollek, ber, kuba, halmai}@inf.u-szeged.hu

‡Department of Trauma Surgery, University of Szeged, H-6720 Szeged, Semmelweis u. 6. E-mail: varga@trauma.szote.u-szeged.hu

was in 1995 for the placement of lumbar pedicle screws [8]. After, this area started to develop rapidly and nowadays there are commercial systems available for solving such surgery problems (see [5], for example). There is a system [9] which focuses on image segmentation, but it has a finite element analysis (FEA) option and navigation modules.

Most of the existing systems are not capable of performing mechanical simulations or are not designed to be used by a surgeon. They have complicated user interfaces meant for an experienced computer user, but surgeons are not usually that experienced. A more user-friendly user interface is required.

Our goal here is to develop a suitable CAS package that is capable of performing biomechanical tests almost in real time and help in the diagnosis of patients who have sustained bone damage. The program can simulate possible surgical solutions and calculate their results. It can also present different stress effects.

This paper describes our ongoing work on computer aided surgical planning and developing a tool for it. In [2] we presented an earlier version of our system, which could perform virtual editing and analysis but had limited capabilities. Our new version has more options and is more user-friendly.

2 Description of the System

The parts of our MedEdit system intended to help the surgeon at the stage of surgical planning are shown in Fig. 1.

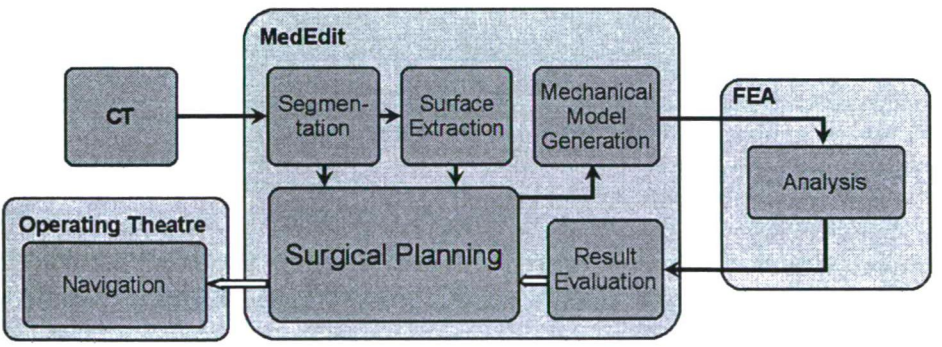


Figure 1: The system schema with its key components and connections. Filled arrows and empty arrows represent data flow and visual feed-back, respectively

Three light grey boxes can be seen here which are the main components of the system. In the MedEdit box there are smaller boxes which we will call modules. For input Computer Tomography (CT) images are available in DICOM (Digital Imaging and Communications in Medicine) format - a common medical imaging standard. MedEdit first reads these images and then stores them in the memory. In

order to generate a mechanical model of the bones, the CT images are segmented, i.e. the pixels representing the bone tissue are separated from the others.

After segmentation information about the object is stored in two data structure formats. The first represents segmented image data as a volume, while the second represents the surface of the data as a triangular mesh. Then the surgeon can perform a virtual operation on the model (the virtual object) by joining broken bone parts, drilling cylindrical holes and inserting screws or implants into the virtually drilled holes with the Surgical Planner module.

By adding material properties to our raw/geometric model in the mechanical model generator a mechanical model is created which can be used to perform a Finite Element Analysis (FEA). Usually using an FEA program requires special engineering knowledge of the material properties and it often has a complex command line user interface. Hence it is really hard to configure and work with. With our system the surgeon can use an FEA program without needing any special expertise. This is because we designed a graphical, mouse-driven user interface in cooperation with surgeons from the local Traumatology Department.

The stress analysis is done using a FEA program and the results are presented in MedEdit. Depending on the results, the surgeon can verify his strategy, or work out a new plan by starting the procedure again from the virtual surgical operation step. It is possible to test different options and analyse the consequences in a virtual environment. Currently we are working on a surgery navigation component, which is in the experimental phase. This could be of use to a surgeon inside the operating theatre after he has planned the operation.

3 Surgical planning in MedEdit

The MedEdit program has a simple modular design, each of the modules performing a special task. The first module imports the DICOM images and segments the bone from the grey scale CT scans. Then a 3D structure is constructed from the segmented volume model. Usually we get a very complex geometrical model, so we use a mesh simplification algorithm to reduce the complexity of the surface. In the fourth module we created a medical surgery planner where the surgeon can try out several possible surgical solutions before performing the actual operation. We implemented various kinds of 3D editing functions like implant insertion, drilling, and slicing. The surgeon can apply forces to the model in the mechanical model generator and then export the data to the FEA system.

3.1 Segmentation

CT scans represent data as greyscale values, these values depending on the absorption of the tissues. Since the bone tissue density differs significantly from other materials like blood and muscle, it is usually possible to separate them. This means that we have a 3D greyscale volume and we create a 3D binary volume where the bone and background voxels are represented by binary-valued data (ones and zeros).

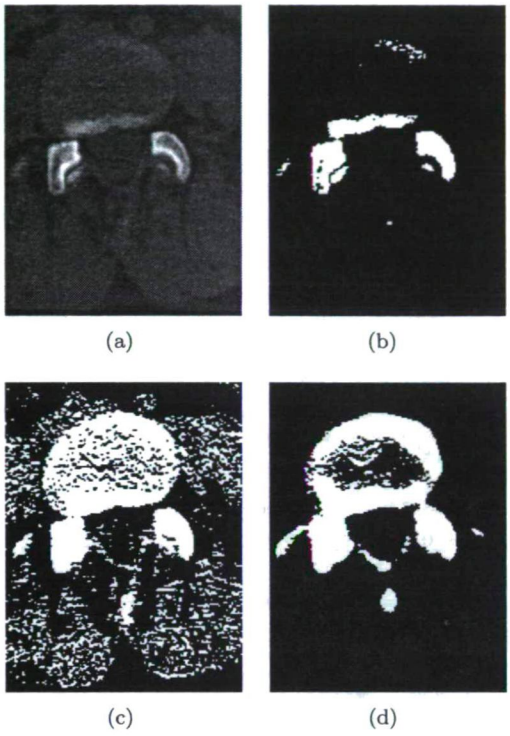


Figure 2: A typical situation when fuzzy segmentation achieves better results than global thresholding. Fig. (a) is the original image of the spine, while Fig. (b) shows the same image with a high threshold value. In this case the bone part is only partially visible. Fig. (c) shows the results with a low threshold value, with a lot of noise appearing around the bone. Fig. (d) shows the same image, but it was segmented by using a fuzzy algorithm.

At first we tried out different thresholds for the bone segmentation (see Figs. 2(b) and 2(c)), but then our experiences showed that this method was only useful when the bone could be easily separated from the muscles. However, in the case of the pelvis this method did not work because other tissues surrounding the pelvis have similar grey values to bone tissue. So we looked for another segmentation method that is just as easy for the user as the thresholding algorithm is. Later we found that a fuzzy connected 3D image segmentation algorithm [3] is good for our needs and can achieve good results (see Fig. 2(d)). The algorithm requires some seed points in the bone tissue then it segments the whole volume containing points having similar properties as the neighbourhood of the selected points. After the segmentation part post-processing is needed to fill in the cavities present in the slices, because inner surfaces are not of interest to us in later steps (see Fig. 3).

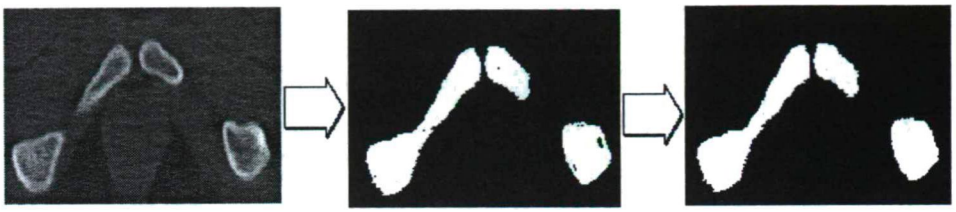


Figure 3: The steps of the segmentation phase. A CT slice showing part of a pelvis (left); the points selected by the fuzzy segmentation algorithm (middle); the results of the post-processing phase (right) after filling in the cavities

3.2 Surface Extraction

For surface extraction and geometry building we tried using contour extraction with slide-by-slide contour simplification followed by a triangularisation [1]. This method works well on tubular shaped objects like a trachea, but fails on complex objects like a human pelvis and requires a lot of effort to convert the data to a triangular format. Later we found that the marching cubes algorithm [6] was ideal for our needs and is more robust. The only drawback of this algorithm is that it invariably produces a high number of triangles (see Table 1).

Table 1: Triangular surfaces generated by the marching cubes algorithm using 512x512 CT scans of different parts of the human body.

Model	Num. of CT slices	Num. of vertices	Num. of faces
Part of a hand	124	219,223	437,256
Part of a knee	109	334,802	668,606
The whole pelvis	91	361,436	721,786

After generating the surface mesh we had to simplify it for several reasons. One was to improve the performance of the rendering engine, and another was because there is a strict upper limit on the number of triangles accepted by our FEA program. To generate a reduced mesh we employed the surface simplification algorithm written by Garland et al. [4] (for an example, see Fig. 4).

3.3 Surgical Planning

In this module the system provides a surgical planner interface for the surgeon. He can plan and simulate several surgical procedures and can check the FEA results as well. It is possible to assemble the broken bone parts by dragging and moving them with the mouse. We use collision detection to simulate the real life behaviour of bones (Fig. 5).

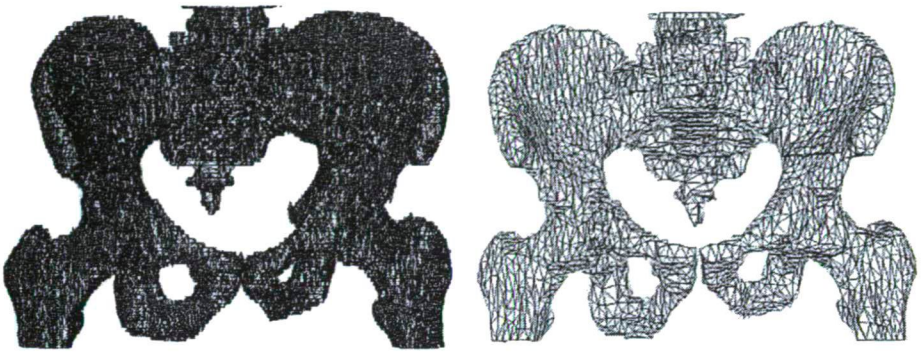


Figure 4: Geometric model and simplification of the pelvis. On the left the model consists of more than 700,000 triangles. On the right the same pelvis after surface simplification (about 10,000 triangles).

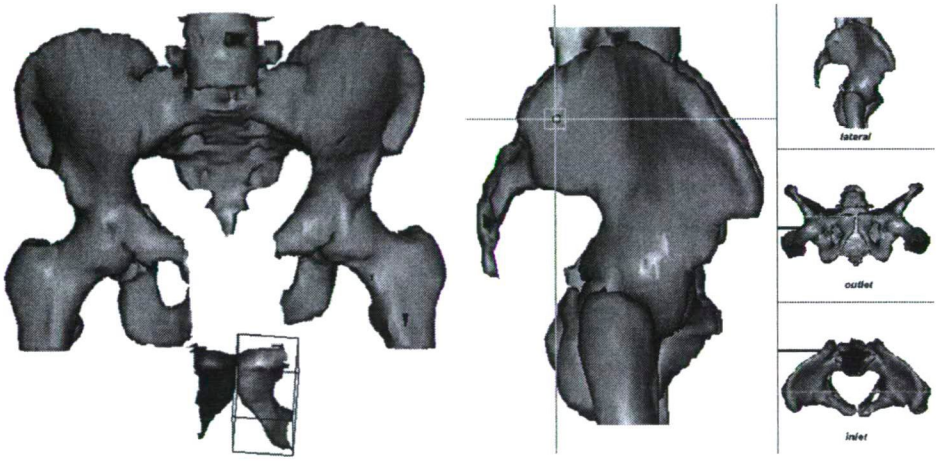


Figure 5: Assembling the simulated bone parts using collision detection (left). The selected bone part is surrounded by its bounding box and can be moved around in the virtual environment. If a collision occurs the other bone is highlighted. On the right, surgical planning using a screw positioning tool and orientation images.

When all the bones are in their positions, virtual implants can be used to hold them together. For example, a hole can be drilled into the bone then a screw implant can be inserted into the model. For the exact positioning the user interface provides various ways of selecting the most appropriate view, angle, and magnification for the presentation of the 3D object (Fig. 5). In order to visualise the complex structures, the surgeon even has the possibility of defining transparent bone properties so that it will look like X-ray transparent bone tissue.

3.4 Mechanical Model Generation

To be able to export the surgical plan to the FEA system, the geometric model has to be extended to include information about material properties, load and boundary conditions. The geometric information is turned into a finite element mesh, and material properties are assigned to the finite elements which form the mesh. Nodal load vectors are set by the user to indicate load effects for certain areas of the mesh and they are then exported to the FEA system in an appropriate file format. Boundary conditions are set for the nodes, which can be fixed or move only in certain directions during the analysis. Our MedEdit system provides a user-friendly interface for defining all these properties.

4 Preparing Surgical Operations for Finite Element Analysis

For the generation of the finite element mesh we have two input data sources: the surface geometry and the segmented 3D volume. The former is treated by the FEA system as a shell, the latter as a solid body. MedEdit can export both of them for the analysis phase.

4.1 Shell Model

When exporting the surface geometry, so-called 3-node shell elements (see Fig. 6) are inserted into the finite element mesh for each surface triangle. This mesh provides the basis for the computations of forces, displacements and deformations. The shell elements are triangular thin planes with blending capabilities and constant thickness.

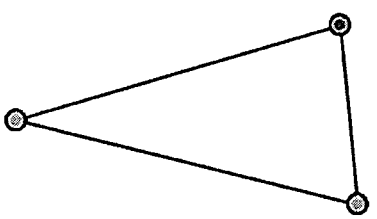


Figure 6: 3-node shell element.

The shell model simulates the outermost 1 mm thick layer of human bones, namely the cortical bone, which is 100 times stiffer than the inner cancelous bone. The material type of the shell elements is given the same physical properties as the cortical bone. Its modulus of elasticity E is 1100 MPa and Poisson's ratio is 0.3. These values are based on cadaver studies [14] and medical literature [15].

Unfortunately adding implants to the virtual model is difficult because it would allow unwanted movements. For example, a drill makes contact only at the surface and the tip could move in undesired directions. We should make it move as it does in real life.

The shell model was used to simulate some typical fractures of the pelvis to get a better understanding of the fracture mechanism. In the next two sections we will briefly describe the way the system handles some of the frequently occurring fracture patterns.

4.1.1 Lateral Compression Fracture

The lateral compression fracture of the pelvic girdle is one of the most common fracture patterns. It can occur when a big force acts on the side of the pelvis. This can happen when a motorcycle rider is hit by a car from the side at a road junction. A typical X-ray image of such an injury is depicted in Fig. 7(a), where the white arrows show the actual fracture.

Our mechanical model of this case is illustrated in Fig. 7(b). The right side of the pelvis is fixed and a force is applied from the patient's left side. In Fig. 7(c) the results of the analysis are shown.

The magnitude of the computed deformation is usually very small. Hence the nodal displacement vectors are multiplied by a suitable scaling factor to visualise the deformation. The details of the displacements are blown up so as to be clearly visible.

4.1.2 Vertical Shear Fracture

Vertical shear injuries typically occur as a result of a fall from a height, but they can also occur in motor vehicle collisions. This force puts an asymmetric axial load on the pelvis, which may result in rotational and vertical instability.

In our simulation the so-called LIV.-LV. vertebrae were fixed and a force was applied to the left femur, as shown in Fig. 8(b). The results of the analysis, which are given in Fig. 8(c), predict the same areas for a fracture as in a typical X-ray image and it also matches clinical expectations.

4.2 Solid Model

When exporting the segmented 3D volume, 8-node solid elements (cube-like elements: see Fig. 9) are inserted into the mesh for each bone-labeled voxel. We use a simple sub-sampling algorithm to reduce the number of voxels and hence keep the computational resources (time and memory) to an acceptable level. Even in the reduced case, a model with about 100,000 elements takes approximately 30 minutes to analyse. This is considerably slower than the shell model where the analysis is completed in a few seconds.

Implants which are placed into the geometrical model during a virtual operation are recalculated and inserted into the 3D volume. This way, implants will be added to the finite element mesh as well (see Fig. 10).

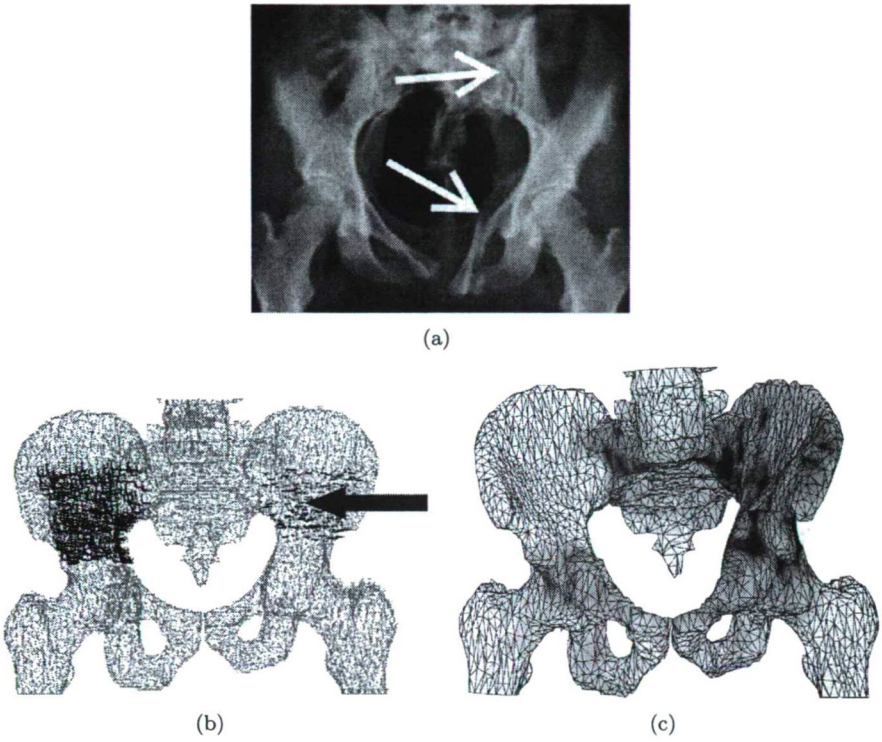
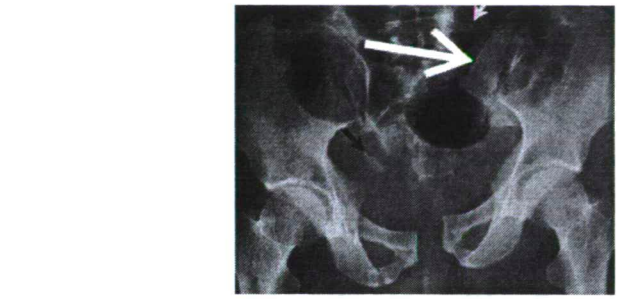


Figure 7: Simulation of the lateral compression of the pelvis. A typical X-ray image of a lateral compression fracture is shown in Fig. (a) above. Fig. (b) shows the mechanical model of the pelvis, the dark area on the patient's right side indicating the points which are fixed, while the arrow on the patient's left shows the direction of the applied force. In Fig. (c) the results of the analysis are presented. Dark areas indicate high material stress which are precisely the areas where a fracture may occur.

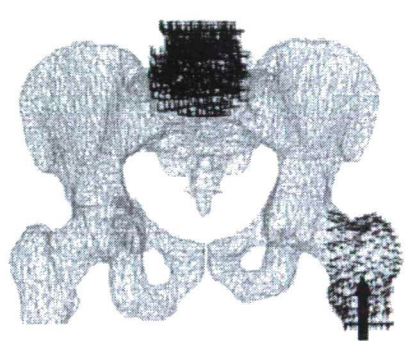
Here the material properties of the model are given average bone values i.e. a modulus of elasticity E of 300 MPa and Poisson's ratio of 0.2.

4.2.1 Knee study

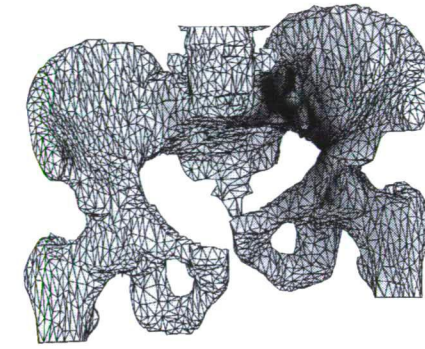
Several tests were performed to compare two different knee fixation techniques. The knee joint of a healthy human was scanned and then imported into our system. With the editing capabilities of MedEdit one artificial fracture was created by deleting bone-labeled voxels from the segmented volume. For the fixation of the simulated fracture two different methods were investigated: internal T-shaped plating fixation, and hybrid external fixation.



(a)



(b)



(c)

Figure 8: Simulation of a vertical shear of the pelvis. Fig. (a) shows a typical X-ray image of a vertical shear fracture. Fig. (b) depicts the mechanical model of the pelvis. The dark area in the spine region indicates the points which are fixed and the arrow on the patient's left shows the direction of the applied force. The results of the analysis are presented in Fig (c).

The mechanical model was created using 8-node hexahedral elements, a load was applied on the top nodes of the knee, and the bottommost nodes were fixed. The results of the analysis (see Fig. 10) confirm that for the internal plate fixation the implant will have high material stress near the line of the fracture. In the case of the other implant, the load is distributed along the whole fixation ring, reducing the risk of a fatigue fracture in the implant material.

4.2.2 Wrist study

A similar study as above was performed on data obtained from a scanned healthy human wrist. One artificial fracture was created on the radius, then both fixation methods were applied in the same way as we did in the knee study. The results obtained (see Fig. 11) have a similar pattern as those obtained in the knee study.

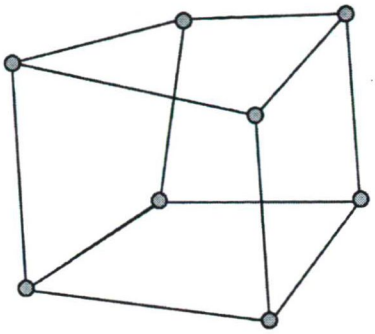


Figure 9: 8-node solid element.

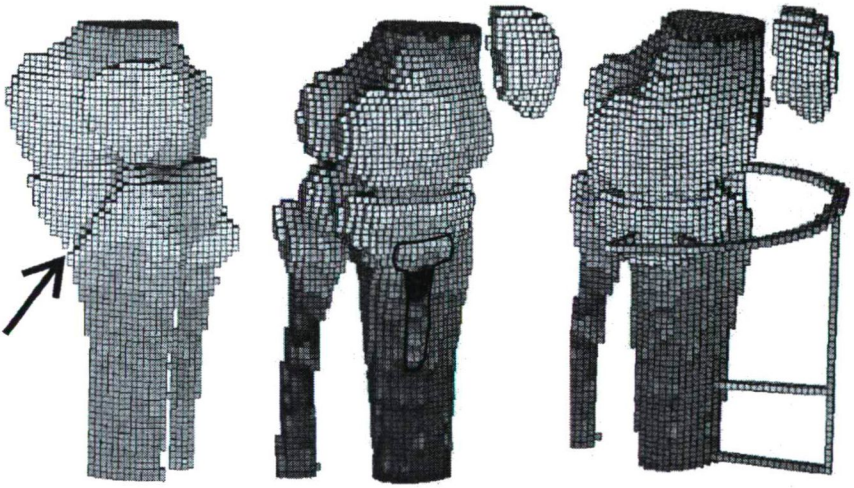


Figure 10: The results of a knee study with two different fixation types. Left, the front view of a knee joint with a simulated fracture (indicated by the black arrow). In the middle, the same knee joint viewed from the right with an interior T-shaped fixation plate. On the right, an external fixation ring is used to stabilise the broken bone.

4.3 Mixed model

Since the solid model requires a lot of computational resources, we developed a mixed model that combines the positive features of the previous two models: one that is small in size, has inner elements, and differentiates between different bone types.

First, using the segmented 3D volume representation and the marching cubes

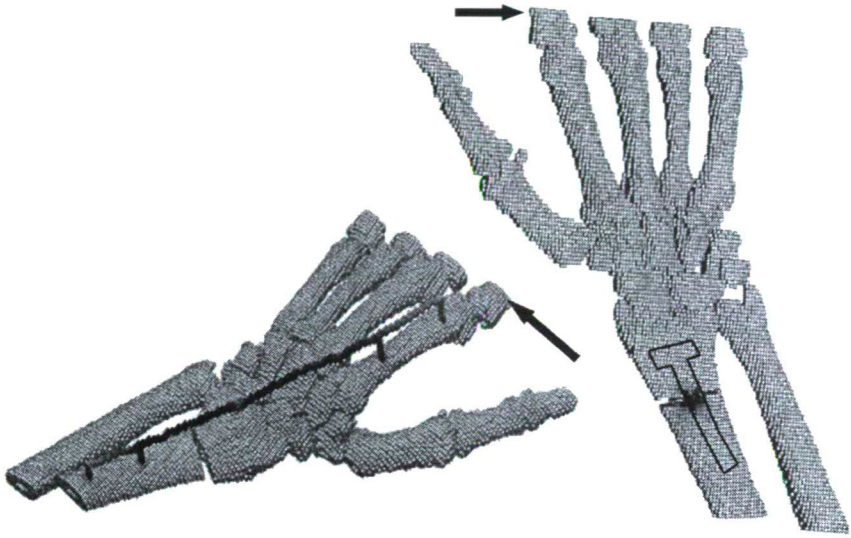


Figure 11: Results of an analysis of a wrist with an artificial fracture after applying an external fixation (left), and applying a T-shaped plate fixation (right).

algorithm, a triangular mesh is created. Each such triangle corresponds to a 3-node shell element in the finite element mesh. Next, 2-node finite elements are added to the mesh to connect the nodes inside the model. From each node rays are cast parallel to the three co-ordinate axes and a matching node is looked for at the intersection on the other side of the model. If a 2-node element is too long, interior points are added. The 2-node elements behave like springs and simulate the inner bone structure, thus preventing the implants from moving in an undesired way.

To see how well it works, we applied the mixed model to a knee study. The results are shown in Fig. 12.

5 Navigation

We are extending our system to help the surgeon find the right points and angles for an implant insertion during an operation. For this reason we started developing a navigation module which is capable of working with three cameras installed in the operating theatre. With this hardware we could use it to identify some special marked points and provide real-time information about where and at which angle the surgeon should insert the implants. This part of the system is now being developed by our team.

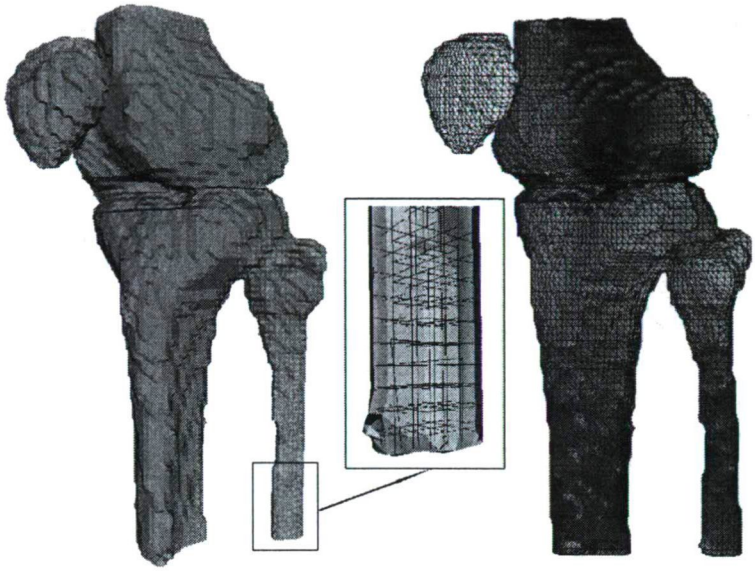


Figure 12: On the left, the mechanical model of the knee joint. The finite element mesh was generated in the same way as that described earlier. The outer 3-node elements are made transparent so that the inner 2-node elements can be seen. On the right, the results of the analysis after applying the same load and boundary conditions as those used in the previous knee study.

6 Summary and Results

In this paper we presented a novel system to help the surgeon in planning orthopedic operations. Using this system we made a virtual biomechanical lab and carried out various FEA studies of a pelvis, a knee, and a wrist. We compared three different finite element mesh types and discussed their advantages and disadvantages. After we created a mixed model and then, using a knee study, we showed that it was faster and requires less memory than the two other models.

The system at present is still in its experimental stage. It is able to perform all the above-mentioned tasks, but there are still parts where some user input or a user decision is required. For example, in the segmentation phase the seed points have to be set manually, and the results should be checked visually to see whether the segmentation meets the user's needs. Then, of course, information exchange with the FEA program is not automatic here. A session file is created by the user during the mesh-generation phase, which will then be read by the FEA program in the analysis phase.

As we said earlier, the MedEdit system has now been implemented and works quite well. In general, it is able to create both the geometric and mechanical models

in about 5 minutes, including the time spent on the manual segmentation. As for the FEA program, it takes about 10 minutes to perform a 3D pelvis volume study (on a PC with a 2-GHz processor and a 1.5 GB RAM).

Our stress results seem to meet clinical expectations, although quantitative tests and measurements still have to be done.

7 Future Plans

As we mentioned in the introduction, the orthopedic-trauma surgeon can generally use only X-ray, CT, MRI images and his own clinical expertise for therapy planning. Mechanical modeling can provide reliable data that ensures the stability of the patient's osteosynthesis prior to the surgical procedure. Certain complications could be avoided if a biomechanical computer modeling approach like ours is used. A detailed and prompt assessment of the surgical problem could be provided with the help of our system.

Our software solution will offer new possibilities for the surgeon. It complements current visual analysis methods and it could also be utilised as a tool in postgraduate education and medical training. We have also created a trainer tool to help students better understand the causes of various bone fractures. Students who are interested can access this system by using our MedSys website [16].

References

[1] Geiger B. *Three-dimensional modeling of human organs and its application to diagnosis and surgical planning.* INRIA, 1993.

[2] Varga E, Halmai Cs, Kuba A, Ollé K, Erdőhelyi B. *MedEdit, a Computer Assisted Planning System for Orthopedic-Trauma Surgery.* Proceedings of the 25th International Conference on Information Technology Interfaces, 2003. p. 507-512.

[3] Nyúl LG, Falcao AX, and Udupa JK. *Fuzzy-Connected 3D Image Segmentation at Interactive Speeds.* SPIE Medical Imaging, 2000. 3979 p.212-23.

[4] Garland M and Heckbert PS. *Surface simplification using quadric error metrics.* ACM SIGGRAPH' Computer Graphics, 1997. p. 209-216.

[5] Robb RA, Hanson DP, and Camp JJ. *Computer-aided surgery planning and rehearsal at Mayo Clinic.* Computer, 1996. 29(1):39-47.

[6] Lorensen WE and Cline HE. *A High Resolution 3D Surface Construction Algorithm.* ACM SIGGRAPH Computer Graphics, 1987. 21, (4).

[7] Galloway RL, Maciunas RJ. *Stereotactic neurosurgery.* Crit. Rev. Biomed. Eng. 18, 1990. 207-233, (26).

[8] Nolte LP, Zamorano LJ, Jiang Z, Wang Q, Langlotz F, Berlemann U. *Image-guided insertion of transpedicular screws: a laboratory set-up.* Spine 1995 497-500, (20).

[9] Mimics at www.materialise.be

[10] Flashpoint 5000 Stryker/Leibinger (www.boulderinnovatorsgroup.com)

[11] NDI Optotrak Northern Digital (www.ndigital.com)

[12] ART/Qualisys ART (www.ar-tracking.de)

[13] Ascension Laser Bird (www.ascension-tech.com)

[14] Varga E.: *Biomechanical and Clinical Investigation of Pelvic Ring Injuries.* Annals of the Albert Szent-Györgyi Medical and Pharmaceutical Center, 2000. Vol. 47.

[15] Beer FB and Johnston ER. *Mechanics of Materials.* McGraw-Hill Inc., 1980. 42.

[16] Our MedSys system website: http://www.inf.u-szeged.hu/~ medsys

Received February, 2004

Acta Cybernetica **17** (2006) 605–614.

ϵ-*Sparse Representations*: Generalized Sparse Approximation and the Equivalent Family of SVM Tasks

Zoltán Szabó* and András Lőrincz†

Abstract

Relation between a family of generalized Support Vector Machine (SVM) problems and the novel ϵ-sparse representation is provided. In defining ϵ-sparse representations, we use a natural generalization of the classical ϵ-insensitive cost function for vectors. The insensitive parameter of the SVM problem is transformed into component-wise insensitivity and thus overall sparsification is replaced by component-wise sparsification. The connection between these two problems is built through the generalized Moore-Penrose inverse of the Gram matrix associated to the kernel.

1 Introduction

Girosi [3] has shown the equivalence of the classic Support Vector Machine (SVM) regression and the sparse approximation scheme [6], similar to the Basis Pursuit De-Noising algorithm [2] under the assumption of noiseless observation. The novelty of the approach is that the approximation is introduced directly in the Reproducing Kernel Hilbert Space (RKHS) and thus it avoids the empirical estimation of the estimation error. Equivalence is understood in the sense that the two optimization problems give rise to the same Quadratic Programming (QP) task.

Equivalence can be shown similarly to [3], but under the condition of noisy observation for linear and quadratic ϵ-insensitive SVM approximation costs [5]. The noise process was included into an extended RKHS. In both cases, however, the ϵ of the approximation cost is transformed onto the scalar multiplier of the parameter vector, which determines the linear combination in the approximation. We ask (i) if it is possible to embed the insensitivity parameter into a constraint on the searched representation, i.e, directly into the cost function, and (ii) if there is an extension of the SVM problems characterized by pair (C, ϵ) (where C is the

*Department of Information Systems, Eötvös Loránd University, H-1117 Budapest, Hungary; E-mail: szzoli@cs.elte.hu

†Corresponding author; Department of Information Systems, Eötvös Loránd University, H-1117 Budapest, Hungary; E-mail: andras.lorincz@elte.hu

multiplier of the ϵ-insensitive cost term of the cost function [12, 3]) to more general problems favoring sparse coding.

The paper is constructed as follows: Section 2 is about the notations and definitions used throughout this work. In Section 3 we sketch earlier correspondences between sparse coding and SVM. Section 4 defines the two generalized problem classes, ϵ-sparse problem class and the corresponding SVM problem class. These classes will be transformed onto each other in this section. Conclusions are drawn in Section 5.

2 Notations and Basic Concepts

For the sake of clarity, our notations and the basic concepts are provided.

2.1 Letter Types, Number Sets

Numbers (b), vectors[1] ($\mathbf{b}$), and matrices ($\mathbf{B}$) are distinguished from each other by letter types. Natural number sets are represented by $\mathbb{N}$, that is, $\mathbb{N} := \{0, 1, 2, \dots\}$, whereas $\mathbb{R}$ stands for real numbers. Subsets restricted for positive values are indicated by $+$ sign, e.g., $\mathbb{N}^+$ and $\mathbb{R}^+$.

2.2 Vectors and Matrices

Relations concerning vectors (e.g.,: $\geq$) are to be meant for each coordinate separately. The i^{th} component of vector $\mathbf{v}$ is denoted by v_i, the ij^{th} component of matrix $\mathbf{V}$ by $V_{i,j}$. ϵ-insensitive cost of vectors is defined as

$$\|\mathbf{v}\|_{\mathbf{r}} := \sum_i |v_i|_{r_i},$$

where $|v|_r := \{0, \text{ if } |v| \leq r; |v| - r, \text{ otherwise}\}$ is the usual 'ϵ-insensitive' cost function[2], which is shown in Fig. 1.

Operations $\mathbf{v}^T$, $\mathbf{v} \circ \mathbf{z}$, and $\mathbf{V} \otimes \mathbf{Z}$ represent transposition, multiplication by elements and the Kronecker product, respectively. Symbol $\mathbf{1}$ has special meaning, it represents a vector having only 1s, i.e., $\mathbf{1} := [1, \dots, 1]^T$.

The *Moore-Penrose generalized inverse* of matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$ is a unique matrix $\mathbf{G}^- \in \mathbb{R}^{m \times n}$, which has the following features:

$$\mathbf{G}\mathbf{G}^-, \mathbf{G}^-\mathbf{G} : \text{symmetric matrices} \quad (1)$$

$$\mathbf{G}\mathbf{G}^-\mathbf{G} = \mathbf{G} \quad (2)$$

$$\mathbf{G}^-\mathbf{G}\mathbf{G}^- = \mathbf{G}^-. \quad (3)$$

[1] Vector means column vector.

[2] Notice that $\mathbf{r} \equiv \mathbf{0}$ gives rise to the L_1 norm for vectors.

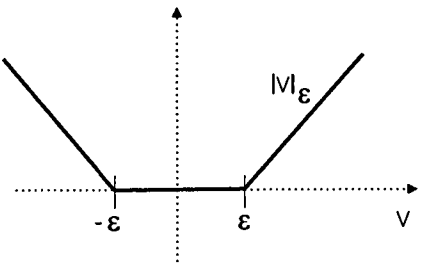


Figure 1: Vapnik's $|v|_\epsilon$ ϵ-insensitive cost function. One may think of this cost function that it represents a resolution not better than ϵ and errors smaller than ϵ are not detected and give rise to no cost. Errors larger than ϵ are, however, detected and – for mathematical tractability – make linear contributions to the cost function.

2.3 RKHS, Feature Mapping, Gram Matrix

Here, we review some basic properties of Reproducing Kernel Hilbert Spaces (RKHS), necessary for our considerations. For further details, the interested reader is referred to the literature [12, 11, 1, 4].

An RKHS is denoted by $\mathcal{H}$. We shall select functions from this space to approximate sample points $\{\mathbf{x}_i, y_i\}_{i=1..l}$, where $\mathbf{x}_i \in \mathcal{X}$ form the *input space* and $y_i \in \mathbb{R}$ (see, e.g, [7]). In space $\mathcal{H}$, the scalar product is computed by means of *kernel* k. Kernel k is also used to define the basic functions of the RKHS: $\phi(\mathbf{x}) := k(\cdot, \mathbf{x}) : \mathcal{X} \to \mathcal{H}$. Such functions are called *feature mappings* and function $\phi(\mathbf{x})$ is interpreted as the *representation* of $\mathbf{x}$ in space $\mathcal{H}$. Now, the scalar product of feature mappings is defined as

$$\langle \phi(\mathbf{s}), \phi(\mathbf{t}) \rangle_{\mathcal{H}} = \langle k(\cdot, \mathbf{s}), k(\cdot, \mathbf{t}) \rangle_{\mathcal{H}} = k(\mathbf{s}, \mathbf{t}) \quad (\mathbf{s}, \mathbf{t} \in \mathcal{X}). \tag{4}$$

It can be shown that the kernel satisfies the following *reproducing property*

$$\langle f(\cdot), k(\cdot, \mathbf{t}) \rangle_{\mathcal{H}} = f(\mathbf{t}) \quad (\mathbf{t} \in \mathcal{X}, \forall f \in \mathcal{H}). \tag{5}$$

This means that $k(\cdot, \mathbf{t})$ can be seen as the *evaluation functional* at position $\mathbf{t}$ of space $\mathcal{H}$. The *Gram matrix of* k defined by $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ l-tuples assumes the following form

$$\mathbf{G} := [G_{i,j}]_{i,j=1\ldots l} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1\ldots l}. \tag{6}$$

2.4 SVM

Function approximation based on sparse data is often hard and is typically ill-posed [4]: existence, uniqueness and stability conditions may not be met in these cases. Regularization theory [10] can be of help under these conditions. To solve such problems, Vapnik, in his pioneering works, formulated the Support Vector Machine

(SVM) problem family [12, 11]. In the SVM problem, the approximating functions are searched in the form

$$f_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle_{\mathcal{H}} + b, \tag{7}$$

subject to ϵ-insensitive cost function

$$V(u,z) = |u-z|_{\epsilon}, \tag{8}$$

and with regularizer [10] of the form $\|\mathbf{w}\|_{\mathcal{H}}^2$ with norm $\|\cdot\|_{\mathcal{H}}$ defined by kernel k of RKHS $\mathcal{H} = \mathcal{H}(k)$. Then the SVM task is as follows:

$$\min_{\mathbf{w},b} H[\mathbf{w},b] := C\sum_{i=1}^{l} |y_i - f_{\mathbf{w},b}(\mathbf{x}_i)|_{\epsilon} + \frac{1}{2}\|\mathbf{w}\|_{\mathcal{H}}^2 \quad (C>0). \tag{9}$$

Optimization of Eq. (9) can be executed, e.g., by solving a Quadratic Programming (QP) task formulated in the dual space

$$\min_{\mathbf{d}^*,\mathbf{d}} \left[\frac{1}{2}(\mathbf{d}^*-\mathbf{d})^T \mathbf{G}(\mathbf{d}^*-\mathbf{d}) - (\mathbf{d}^*-\mathbf{d})^T \mathbf{y} + (\mathbf{d}^*+\mathbf{d})^T \epsilon \mathbf{1}\right] \tag{10}$$

$$\text{provided that} \left\{ \begin{array}{rcl} C\mathbf{1} \geq \mathbf{d}^*, \mathbf{d} & \geq & \mathbf{0} \\ (\mathbf{d}^*-\mathbf{d})^T \mathbf{1} & = & 0 \end{array} \right\}.$$

For the derivation, see, e.g., [9]. Here, matrix $\mathbf{G}$ is the Gram matrix introduced before.

3 Previous Results

3.1 Noiseless Case

Starting from the work [2] Girosi has formulated a modified sparse approximation task in RKHS [3]:

$$\min_{\mathbf{a}} \left[\frac{1}{2}\left\| f(\cdot) - \sum_{i=1}^{l} a_i k(\cdot, \mathbf{x}_i)\right\|_{\mathcal{H}}^2 + \epsilon \|\mathbf{a}\|_1\right]. \tag{11}$$

The first term is about quadratic approximation but instead of $\mathbb{R}$ it is formulated through the norm $\|\cdot\|_{\mathcal{H}}^2$ on Hilbert space $\mathcal{H}$. The second term is the sparse constraint, or sparsifying cost term. Girosi has shown that Eq. (11) is equivalent to the SVM task of Eq. (9) provided that

1. objective f is in $\mathcal{H}$ and that $\langle f, 1\rangle_{\mathcal{H}} = 0$,[3]

[3] This restriction gives rise to constraint $\sum_i a_i = 0$.

2. data are noise-free, that is $f(\mathbf{x}_i) = y_i$ $(i = 1, \ldots, l)$,

3. $C \to \infty$.

Equivalence is to be understood in the sense that by breaking the searched vector $\mathbf{a}$ into positive and negative parts, such as

$$\mathbf{a} = \mathbf{a}^+ - \mathbf{a}^-, \text{ where } \mathbf{a}^+, \mathbf{a}^- \geq \mathbf{0}, \text{ and } \mathbf{a}^+ \circ \mathbf{a}^- = \mathbf{0} \tag{12}$$

then the task for pair $(\mathbf{a}^+, \mathbf{a}^-)$ is identical to the optimal solution $(\mathbf{d}^*, \mathbf{d})$ for Eq. (10) in the dual QP space.

3.2 The Noisy Case

The solution was extended to the noise case [5]: the connection was formulated for the regression problem and for linear and quadratic ϵ-insensitive SVM approximation. The equivalence is based on a larger RKHS space, which encapsulates the noise process, too. For detailed description and for other similar equivalences, the interested reader is referred to the original work [5].

In the cited cases [3, 5], the insensitive parameter (ϵ) was transformed into the multiplier of the sparsifying cost term (compare, e.g., Eq. (9) and Eq. (11)). Our question is if the constant multiplier of the ϵ-insensitivity loss can be transformed directly into the different components of the loss function by generalizing uniform sparsification to a component-wise sparsification problem.

For notational simplicity, instead of approximating in semi-parametric form (e.g., $f + b$, where $f \in \mathcal{H}$), we shall deal with the so called non-parametric scheme [8] ($f \in \mathcal{H}$). This approach is well grounded by the representer theorem [8].

4 Generalized Problems

In this section we shall introduce the generalizations of the previous SVM and sparse tasks and we shall show that they are equivalent. Given this equivalence, the two problem family will be referred jointly as *ϵ-sparse representations.*

4.1 The (c, e)-SVM Task

Below, we introduce an SVM task family, which can be connected to regularization theory and satisfies the conditions of the representer theorem [8]. The usual SVM task – Eq. (9) – is modified as follows:

1. We shall approximate in the form $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle_{\mathcal{H}}$. The representer theorem warrants that it is satisfactory to approximate in this special form from $\mathcal{H}$.

2. We shall use approximation errors that may differ for each sample point.

3. We shall use weights that may differ for each sample point.

Introducing vector $\mathbf{e}$ for the ϵ-insensitive costs and $\mathbf{c}$ for the weights, respectively, the generalized problem has the following form:

$$\min_{\mathbf{w}} \left[\sum_{i=1}^{l} c_i \left| y_i - f_{\mathbf{w}}(\mathbf{x}_i) \right|_{e_i} + \frac{1}{2} \|\mathbf{w}\|_{\mathcal{H}}^2 \right] \quad (\mathbf{c} > \mathbf{0}, \mathbf{e} \geq \mathbf{0}). \tag{13}$$

This task shall be called the $(\mathbf{c}, \mathbf{e})$-*SVM task*. The original task of Eq. (9) corresponds to the particular choice of $((C, \epsilon) \otimes \mathbf{1})$ and $b = 0$. Alike to the original SVM task, the new $(\mathbf{c}, \mathbf{e})$-SVM task also has its quadratic equivalent in the dual space, which is as follows

$$\min_{\mathbf{d}^*, \mathbf{d}} \left[\frac{1}{2} (\mathbf{d}^* - \mathbf{d})^T \mathbf{G} (\mathbf{d}^* - \mathbf{d}) - (\mathbf{d}^* - \mathbf{d})^T \mathbf{y} + (\mathbf{d}^* + \mathbf{d})^T \mathbf{e} \right], \tag{14}$$

$$\text{provided that } \{ \ \mathbf{c} \geq \mathbf{d}^*, \mathbf{d} \ \geq \ \mathbf{0} \ \},$$

where $\mathbf{G}$ denotes the Gram matrix of kernel k that belongs to points $\mathbf{x}_i$.

4.2 The $(\mathbf{p}, \mathbf{s})$-Sparse Task

Let us consider the optimization problem

$$\min_{\mathbf{a}} F[\mathbf{a}] := \frac{1}{2} \left\| f(\cdot) - \sum_{i=1}^{l} a_i k(\cdot, \mathbf{x}_i) \right\|_{\mathcal{H}}^2 + \sum_{i=1}^{l} p_i \left| a_i \right|_{s_i} \quad (\mathbf{p} > \mathbf{0}, \mathbf{s} \geq \mathbf{0}) \tag{15}$$

on sample points $\{\mathbf{x}_i, y_i\}_{i=1..l}$ that intends to approximate objective function $f \in \mathcal{H}(k)$. This problem shall be referred to as $\mathbf{p}$-*weighted and* $\mathbf{s}$-*sparse task*, or $(\mathbf{p}, \mathbf{s})$-*sparse task*, for short. The particular choice of $((\epsilon, 0) \otimes \mathbf{1})$ recovers the sparse representation form of Eq. (11).

4.3 Correspondence Between the Tasks

The tasks defined by Eq. (13) and Eq. (15), respectively will be connected to each other by means of the following theorem:

Theorem 1. Let $\mathcal{X}$ denote and arbitrary non-empty set, k be a kernel on $\mathcal{X}$, $\{\mathbf{x}_i, y_i\}_{i=1..l}$ a sample set of l elements, where $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathbb{R}$. Assuming that the values of RKHS objective $f \in \mathcal{H} = \mathcal{H}(k)$ can be observed in points $\mathbf{x}_i$ $(f(\mathbf{x}_i) = y_i)$, then under the approximation

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle_{\mathcal{H}}$$

the dual problems of the

$$\min_{\mathbf{w}} \left[\sum_{i=1}^{l} c_i \left| y_i - f_{\mathbf{w}}(\mathbf{x}_i) \right|_{e_i} + \frac{1}{2} \|\mathbf{w}\|_{\mathcal{H}}^2 \right] \quad (\mathbf{c} > \mathbf{0}, \mathbf{e} \geq 0)$$

$(\mathbf{c},\mathbf{e})$-SVM task and that of

$$\min_{\mathbf{a}}\left[\frac{1}{2}\left\|f(\cdot)-\sum_{i=1}^{l}a_i k(\cdot,\mathbf{x}_i)\right\|_{\mathcal{H}}^2+\sum_{i=1}^{l}p_i\,|a_i|_{s_i}\right]\quad(\mathbf{p}>0,\mathbf{s}\geq 0)$$

the $(\mathbf{p},\mathbf{s})$-sparse task can be transformed onto each other through the generalized inverse $\mathbf{G}^-$ of Gram matrix

$$\mathbf{G}:=[G_{i,j}]_{i,j=1\ldots l}=[k(\mathbf{x}_i,\mathbf{x}_j)]_{i,j=1\ldots l},$$

or, shortly,

$$\textit{Dual}\,[(\mathbf{c},\mathbf{e})\text{–SVM}]\overset{\mathbf{G}^-}{\longleftrightarrow}\textit{Dual}\,[(\mathbf{p},\mathbf{s})\text{–sparse}]$$

under correspondence

$$(\mathbf{d}^*,\mathbf{d},\mathbf{G},\mathbf{y})\leftrightarrow(\mathbf{d}^+,\mathbf{d}^-,\mathbf{G}^-\mathbf{G}\mathbf{G}^-,\mathbf{G}^-\mathbf{y})=(\mathbf{d}^+,\mathbf{d}^-,\mathbf{G}^-,\mathbf{G}^-\mathbf{y}).$$

Proof. We shall modify Eq. (15) under the assumption of $f(\mathbf{x}_i)=y_i\quad(i=1,\ldots,l)$. Given that norm $\|\cdot\|_{\mathcal{H}}^2$ is induced by a scalar product on $\mathcal{H}$, and utilizing the bilinear property of scalar products, we have

$$\begin{aligned}F[\mathbf{a}] &= \frac{1}{2}\|f\|_{\mathcal{H}}^2-\sum_i a_i\,\langle f(\cdot),k(\cdot,\mathbf{x}_i)\rangle_{\mathcal{H}}+ \\ &\quad+\frac{1}{2}\sum_{i,j}a_i a_j\,\langle k(\cdot,\mathbf{x}_i),k(\cdot,\mathbf{x}_j)\rangle_{\mathcal{H}}+\sum_i p_i\,|a_i|_{s_i}.\end{aligned}\tag{16}$$

The reproducing property of the kernel can be applied to show

$$\langle f(\cdot),k(\cdot,\mathbf{x})\rangle_{\mathcal{H}} = f(\mathbf{x})=y_i,\tag{17}$$

$$\langle k(\cdot,\mathbf{x}_i),k(\cdot,\mathbf{x}_j)\rangle_{\mathcal{H}} = k(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{G}_{i,j},\tag{18}$$

where the Gram matrix notation was used. Now, neglecting the first term of $F[\mathbf{a}]$, which is independent of $\mathbf{a}$, one has

$$\frac{1}{2}\mathbf{a}^T\mathbf{G}\mathbf{a}-\mathbf{y}^T\mathbf{a}+\sum_i p_i\,|a_i|_{s_i}\to\min_{\mathbf{a}}.\tag{19}$$

Then the s-insensitive terms can be rewritten by introducing slack variables [9] and the following form can be derived

$$\min_{\mathbf{a},\mathbf{a}^+,\mathbf{a}^-}\left[\frac{1}{2}\mathbf{a}^T\mathbf{G}\mathbf{a}-\mathbf{y}^T\mathbf{a}+\mathbf{p}^T\left(\mathbf{s}^+ +\mathbf{s}^-\right)\right],\tag{20}$$

$$\text{provided that}\left\{\begin{array}{rcl}\mathbf{a} &\leq& \mathbf{s}+\mathbf{s}^+\\ -\mathbf{a} &\leq& \mathbf{s}+\mathbf{s}^-\\ \mathbf{0} &\leq& \mathbf{s}^+,\mathbf{s}^-\end{array}\right\},$$

with its dual form given as

$$\max_{\mathbf{d}^+,\mathbf{d}^-,\mathbf{q}^+,\mathbf{q}^- \geq 0} L(\mathbf{d}^+,\mathbf{d}^-,\mathbf{q}^+,\mathbf{q}^-) = \tag{21}$$
$$= \frac{1}{2}\mathbf{a}^T\mathbf{G}\mathbf{a} - \mathbf{y}^T\mathbf{a} + \mathbf{p}^T\left(\mathbf{s}^+ + \mathbf{s}^-\right) - (\mathbf{q}^+)^T\mathbf{s}^+ - (\mathbf{q}^-)^T\mathbf{s}^- -$$
$$-(\mathbf{d}^+)^T(\mathbf{s} + \mathbf{s}^+ - \mathbf{a}) - (\mathbf{d}^-)^T(\mathbf{s} + \mathbf{s}^+ + \mathbf{a}).$$

According to the condition on the saddle-point, the derivatives of Langrangian L taken by the prime variables disappear at optimum, that is

$$\mathbf{0} = \frac{dL}{d\mathbf{a}} = \mathbf{a}^T\mathbf{G} - \mathbf{y}^T + (\mathbf{d}^+ - \mathbf{d}^-)^T, \tag{22}$$
$$\mathbf{0} = \frac{dL}{d\mathbf{s}^+} = \mathbf{p}^T - (\mathbf{d}^+)^T - (\mathbf{q}^+)^T, \tag{23}$$
$$\mathbf{0} = \frac{dL}{d\mathbf{s}^-} = \mathbf{p}^T - (\mathbf{d}^-)^T - (\mathbf{q}^-)^T. \tag{24}$$

Reordering and transposing Eq. (22), we have

$$\mathbf{a}^T\mathbf{G} = \left(\mathbf{y} - (\mathbf{d}^+ - \mathbf{d}^-)\right)^T, \tag{25}$$
$$\mathbf{G}\mathbf{a} = \left(\mathbf{y} - (\mathbf{d}^+ - \mathbf{d}^-)\right), \tag{26}$$

where the symmetric property of Gram matrix $\mathbf{G}$ was exploited. One can replace matrix $\mathbf{G}$ of the Lagrangian by $\mathbf{G}\mathbf{G}^-\mathbf{G}$ according to Eq. (2). Also, considering that

$$\mathbf{a}^T\mathbf{G}\mathbf{a} = \mathbf{a}^T(\mathbf{G}\mathbf{G}^-\mathbf{G})\mathbf{a} = (\mathbf{a}^T\mathbf{G})\mathbf{G}^-(\mathbf{G}\mathbf{a}) \tag{27}$$

one can insert the expressions for $\mathbf{a}^T\mathbf{G}$ and $\mathbf{G}\mathbf{a}$ from Eqs. (25) and (26), respectively. Equations (23) and (24) can also be applied for Lagrangian L. Variables $\mathbf{q}^+, \mathbf{q}^-$ disappear from Lagrangian L, but the non-negativity conditions Eqs. (23) and (24) give rise to constraints $\mathbf{p} \geq \mathbf{d}^+$ and $\mathbf{p} \geq \mathbf{d}^-$ for variables $\mathbf{d}^+$ and $\mathbf{d}^-$. We can also change the minimization of Lagrangian L to maximization by changing the sign.

Taken together, we have the QP task

$$\min_{\mathbf{p} \geq \mathbf{d}^+, \mathbf{d}^- \geq 0} \left[\frac{1}{2}(\mathbf{y} - (\mathbf{d}^+ - \mathbf{d}^-))^T\mathbf{G}^-(\mathbf{y} - (\mathbf{d}^+ - \mathbf{d}^-)) + (\mathbf{d}^+ + \mathbf{d}^-)^T\mathbf{s}\right]. \tag{28}$$

The terms of the quadratic expression can be expanded and reordered. Upon dropping terms not containing variables $\mathbf{d}^+$ or $\mathbf{d}^-$, and making use of the symmetric property of $\mathbf{G}^-$ inherited from $\mathbf{G}$, one has

$$\min_{\mathbf{p} \geq \mathbf{d}^+, \mathbf{d}^- \geq 0} \left[\frac{1}{2}(\mathbf{d}^+ - \mathbf{d}^-)^T\mathbf{G}^-(\mathbf{d}^+ - \mathbf{d}^-) - (\mathbf{d}^+ - \mathbf{d}^-)^T\mathbf{G}^-\mathbf{y} + (\mathbf{d}^+ + \mathbf{d}^-)^T\mathbf{s}\right]. \tag{29}$$

Now, we are in the position to compare this optimization task with Eq. (14) by making use of the generalized inverse $\mathbf{G}^-$ of Gram matrix $\mathbf{G}$. The result is that

$$\text{Dual}[(\mathbf{c},\mathbf{e})\text{–SVM}] \leftrightarrow \text{Dual}[(\mathbf{p},\mathbf{s})\text{–sparse}]\,.$$

In short, we proved that the two tasks transform onto each other through $\mathbf{G}^-$ in the following way

$$(\mathbf{d}^*,\mathbf{d},\mathbf{G},\mathbf{y}) \leftrightarrow (\mathbf{d}^+,\mathbf{d}^-,\mathbf{G}^-\mathbf{G}\mathbf{G}^-,\mathbf{G}^-\mathbf{y}) = (\mathbf{d}^+,\mathbf{d}^-,\mathbf{G}^-,\mathbf{G}^-\mathbf{y}), \qquad (30)$$

where in the last step, property $\mathbf{G}^-\mathbf{G}\mathbf{G}^- = \mathbf{G}^-$ of the generalized inverse (Eq. (3)) was exploited. □

5 Conclusions

We have extended the concept of sparse representation in RKHSs to a larger class of tasks, where individual components can have individual sparsifying terms. We showed that alike to the original sparse formulation, the generalized ϵ-sparse approach also has an equivalent SVM task family. This novel formulation may gain applications in signal processing, clustering and categorization problems.

References

[1] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[2] S.S. B. Chen, D.L. Donoho, and M.A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal of Scientific Computing*, 20(1):33–61, 1999.

[3] F. Girosi. An Equivalence Between Sparse Approximation and Support Vector Machines. *Neural Computation*, 10(6):1455–1480, 1998.

[4] R. Herbrich. *Learning Kernel Classifiers.* The MIT Press, 2002.

[5] S.Y. Huang and Y.J. Lee. Equivalence Relations Between Support Vector Machines, Sparse Approximation, Bayesian Regularization and Gauss-Markov Prediction. Technical report, Inst. of Stat. Science, Academia Sinica, Computer Sci. and Inf. Engn., National Taiwan University, 2003.

[6] B. Olshausen and D.J. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37:3311–3325, 1997.

[7] B. Schölkopf, C.J.C. Burges, and A.J. Smola. *Advances in Kernel Methods - Support Vector Learning.* MIT Press, Cambridge, Ma., 1999.

[8] B. Schölkopf, R. Herbrich, and A.J. Smola. A Generalized Representer Theorem. In *Proc. of the 14th Ann. Conf. on Comp. Learning Theory*, volume 2111 of *Lecture Notes In Computer Science*, pages 416–426, London, UK, 2001. Springer-Verlag.

[9] A.J. Smola. *Learning with Kernels.* PhD thesis, Technische Universität Berlin, 1998.

[10] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-Posed Problems.* Winston, Washington, DC, USA, 1977.

[11] V. Vapnik. *Statistical Learning Theory.* Wiley, Chichester, GB, 1998.

[12] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., 1995.

Received March, 2005

Selected Papers from the *2nd Conference on Hungarian Computational Linguistics*

Preface

This issue of *Acta Cybernetica* contains three papers whose preliminary versions appeared in Hungarian language in the proceedings of the 2nd Conference on Hungarian Computational Linguistics. The conference was held in Szeged on December 9–10, 2004 and its aim was to provide a forum for researchers working on Hungarian computational linguistics and speech processing, see `http://www.inf.u-szeged.hu/mszny2004/`.

After the conference, the authors were invited to submit completed versions of their papers to *Acta Cybernetica*. All submitted papers were then subjected to the normal refereeing process of the journal. Altogether seven manuscripts were submitted, out of which three have been accepted.

We thank the authors and the referees for their help in the preparation of this issue.

The 3rd Conference on Hungarian Computational Linguistics was held also in Szeged on December 8–9, 2005.

Zoltán Alexin
Guest Editor

Acta Cybernetica **17** (2006) 617–632.

Word Order and Discontinuities in Dependency Grammar

C. Bartha*, T. Spiegelhauer†, R. Dormeyer†, and I. Fischer†

Abstract

Natural languages are always difficult to parse. Two phenomena that constantly pose problems for different formalisms are word order—what part of a sentence has to be placed where—and discontinuities—words that belong together but are not placed into the same phrase. Dependency grammar, a linguistic formalism based on binary relations between words, is very adequate for handling both problems. A parser for dependency grammar together with its grammar writing formalism is described in this paper. Word order and discontinuities in Hungarian are handled based on this formalism.

1 Introduction

When taking a look in the standard literature [9] on computational linguistics, long introductions in phrase structure grammars invented by Chomsky can be found. They have been in the focus for nearly fifty years now. Phrase structure grammars turned out to be a helpful method when modeling English; quite a lot of parsers can be found together with extensive grammars. But it also turned out that they are not useful when it comes to languages with free or semi-free word order. Discontinuous constituents and long distance dependencies pose difficulties, too. Several work arounds and extensions have been invented to overcome these problems. Some of these extensions and new developments, e.g. *Head-Driven Phrase Structure Grammar* [13], are similar to dependency grammar. Dependency grammar, invented by Lucien Tesnière [14], [15], is popular in Europe and Japan. In this paper, a parser for dependency grammar [4] is described. Currently grammars are written for several different languages. Grammar fragments for English, German and Latin have been written [4]. These languages differ in their word order. English has a fixed word order, e.g. the subject has to come first in a declarative sentence. In German, the word order is semi-free. For noun phrases, it is fixed; on the sentence level, the verb has to be in the second position in a declarative sentence. Other elements can

*Siemens PSE Hungary, Szeged, CSS IBS5, E-mail: csongor.barta@siemens.com

†Lehrstuhl für Informatik 2, Friedrich–Alexander Universität Erlangen–Nürnberg, Martensstr. 3., 91058 Erlangen, Germany, E-mail: tilly79@gmx.de, ricarda@dormeyer.de, Ingrid.Fischer@informatik.uni-erlangen.de

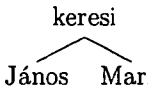


Figure 1: A dependency tree for *János keresi Marit.*

take the other positions, no restrictions are given here. In Latin there are even less word order restrictions. Words can be placed nearly everywhere.

But not only word order is of special interest. Another problem are words that belong together but are not placed next to each other in the sentence. This phenomenon can be found in all languages analyzed. An English example is fronting as in *Ann John told me he had seen.* In German and Hungarian verb prefixes can be separated from the verb and move to another position.

At the moment especially non-indo-European languages are researched. Currently grammars for Japanese [17] and Hungarian are developed. For Japanese, a lot of different dependency grammar implementations exist. This is not the case for Hungarian. Only one international publication could be found containing a dependency grammar for Hungarian [18]. The grammar described in [18] differs a lot from our approach. In [18] first all prefixes and suffixes are separated from word stems. The resulting string is the input for the dependency parser. In our approach this separation does not take place, a sentence is analyzed in its original writing.

In the sequel, our dependency parser and the Hungarian grammar developed up to now are described. In Section 2 the basics of dependency grammar are introduced. After this linguistic introduction, our dependency parser is specified in Section 3. Special features of our Hungarian grammar are given in Section 4. In Section 5 we describe the underlying algorithm. We end with a conclusion.

2 A Short Overview on Dependency Grammar

In dependency grammar binary relations between the words of a sentence are used as the basic construct. The most important part of a sentence is the verb, it opens several slots for other parts of the sentence. Taking the verb *keresi (to seek)*[1] it opens two slots, one for a noun in the nominative case, which is the subject, and one for a noun in the accusative case, the object. In the sentence *János keresi Marit (János seeks Mari)* these slots are occupied by *János*, the subject, and *Marit*, the object. Normally the relations are visualized with the help of trees. A tree for the running example is given in Figure 1.Please note, that every combination of the three words results in the same dependency tree: *János Marit keresi*, *Marit keresi János*, ...

[1]All Hungarian examples used throughout this paper including the English translations are cited from [11]. This example is taken from page 2, example (1).

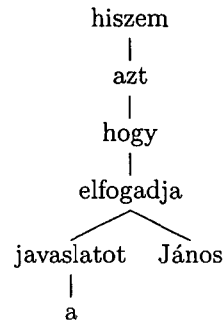


Figure 2: A dependency tree for *János azt hiszem hogy elfogadja a javaslatot. (János, I think, that (he) accepts the proposal.)*

The subject and object can also open new slots. A simple noun opens a slot for e.g. one determiner and any number including zero of adjectives. This means that several different kinds of slots are necessary. First slots are used that must be filled, with one element. If the element is missing, the corresponding sentence is grammatically not correct. In Figure 1 exactly one object is needed. In English each verb needs exactly one subject. Then there are slots that are optional, they can be filled but do not have to be filled. Time and place are optional for most verbs. They can be added, but they can also be left out. Another example for this is the subject in Hungarian. Finally there are slots that can be filled several times, e.g. a noun can take several adjectives. A word opening a slot is also called the head, the word filling the slot will be called the dependent in the rest of the paper.

Long distance dependencies or discontinuous constituents are another phenomenon that pose constant problems to formal grammars and the corresponding parsers. In Hungarian long distance dependencies have been described by different researchers with the earliest publication stemming from the beginning of the last century [11]. In dependency grammar a relation between a head and a dependent is considered discontinuous if not all words between this head and this dependent depend on one of the two. A Hungarian example with discontinuous constituents is given in the following table: [2]

János	*azt*	*hiszem*	*hogy*	*elfogadja*	*a*	*javaslatot*
János	*that-acc*	*I think*	*that*	*accept*	*the*	*proposal*
János, I think that (he) accepts the proposal						

János depends on the verb *elfogadja (accepts).* Between the head *elfogadja* and the dependent *János*, the words *azt hiszem hogy* are found. *hiszem (I think)* as the main verb is the head of the sentence. All other words including *elfogadja (accepts)* and *János* depend somehow on *hiszem. elfogadja* depends of the head *hogy (that)* wich depends directly from *azt (that-acc). azt (that-acc)* finally depends on *hiszem*

[2]This example including the English translation is taken from [11], page 258, example 77.

(I think). Also *azt (that-acc)* and *hogy (that)* form a discontinuity as *hiszem (I think)* in between does not depend on either of the two but *azt (that-acc)* depends on *hiszem.*

The corresponding tree is given in Figure 2. The linear structure of the words in this sentence cannot be reconstructed from Figure 2. In this tree only the syntactic structure is given.

In phrase structure grammar linear and syntactic structure are combined in one tree leading to crossing edges in the phrase structure tree for this sentence. Trees with crossing edges cannot be constructed with context-free grammars.

3 A Parser for Dependency Grammar

Our parser [16] is based on three concepts. The parsing algorithm itself is similar to the well–known Cocke–Kasami–Younger algorithm for context-free phrase structure grammars [9]. The first dependency parser based on this idea was presented in [12]. Words are described by feature structures [9] enriched by a few symbols necessary for dependency grammars. Feature structures are combined with the help of graph unification. Our handling of discontinuous constituents and word order restrictions differs from [12].

3.1 Word Order

In Tesnière's original approach, word order was unimportant for syntactic description. Any order was allowed. This is not useful for parsers, too many wrong sentences would be accepted. The order between head and dependent must be considered as well as the order between the different dependents of one head. Also the number of elements following or preceding a word can be important. E.g. in German the verb in a declarative sentence must fill the second position. In our approach each word has a position list where positions of the word itself and other words are described. This includes as a minimum a position for the word itself. Then each dependent of a word can have a fixed position in this position list. Free positions within the position list are also possible, a free position can take every dependent that is not marked as fixed. The position list makes parsing easier: When a fixed position is following according to this list, the parser has to check for just one element. If the next element is free, only free elements have to be checked.

3.2 Discontinuous Constituents

Linguistically, a discontinuous dependency can be regarded as a ternary relation, i.e. a relation between a dependent, its *syntactic head* and its *linear head* [1]. The syntactic head is the word containing a slot for the discontinuous dependent. But because the syntactic head's constituent is discontinuous, the dependent is positioned in the position list of the linear head. In the example sentence, the syntactic head for *János)* is *elfogadja (accept)* and its linear head is *hiszem (I think).* The

```
Word "Angéla" <"Name"> [
    lexeme: Angéla;
    gender: fem;
    case: nom;]

Word "olvas"   <"VerbPres"> [
    lexeme: olvas;
    mood: declarative;
    number: sing;
    person: 3;]

Template "Name" [
    category: noun;
    special: propername;
    number: sing;
    person: 3;]
```

```
Template "VerbPres" [
    category: verb;
    form: finite;
    tense: present;
    sentence: declarative;
    subj:  oslot [
        category: noun;
        cont: +;
        case: nom;];
    order: (%1 %2 i %3);]
%1 = slot  [cont: +;];
%2 = mslot [cont: +;];
%3 = mslot [cont: +;];
```

Figure 3: Simple grammar with templates for *Angéla olvas (Angéla reads)*

dependent fills a slot of its syntactic head, but occupies a position of the linear head's position list. Because of this, the processing of discontinuous dependencies has to work with linear and syntactic head. The parser must allow for all possible orders between syntactic head, linear head and dependent in a sentence.

3.3 Other Approaches

Over the years several other parsers for dependency grammar have been proposed. In [1] linear order and syntactic order are strictly separated, something we tried to avoid in our approach, because it makes grammar writing less intuitive and parsing less efficient. Fraser's parser [5] is based on backtracking and uses a parsing stack. Covington's approach [2] is cited very often. He invented a simple backtracking algorithm for free word order. But his approach is not well suited for semi-free word order phenomena. Finally there are several dependency parsers based on constraint resolution, a completely different approach [3].

4 Parsing Hungarian

In this section two Hungarian sentences are analyzed. With these examples our grammar description language is described.

4.1 Grammar Description Language and Free Word Order

The grammar description language is important, as it must be easy to learn and to handle for the linguist writing grammars for the parser. We will introduce it with the help of the easy example *Angéla olvas (Angéla reads.)*. In Figure 3, the corresponding grammar is given. Please note that due to space our example grammars are not complete.

To shorten the grammar, templates as introduced by [6] are possible. Templates encode parts of the feature structures that are used very often. Within the entries for words template names are used instead of complete feature structures. As a first step, the lexicon is transformed before parsing. Template names are removed, the feature structure parts these names described are unified with the rest of the feature structure.

Feature structures are started by "[" and ended with "]", features and values are separated by ":" and feature-value-pairs are separated by ";". In Figure 3 two word entries and two templates are given. The lexical entry for *Angéla* contains a template named `Name`, which is also given. Feature structures for `Name` and *Angéla* are unified leading to an entry where agreement features as `number`, `gender`, `case` (not all possible cases are given) and `person` are described. Also the `lexeme` and `category` are shown. The verb *olvas (reads)* is also composed with the template for verbs in present tense. This template is more interesting. It contains an optional slot for a subject indicating that in Hungarian, the subject can be left out. At the end the special feature `order` indicates possible positions. As this position list is used for every word in present tense, it is more complicated than necessary for our small example. The symbol `i` stands for the position of the word itself, in this case the current verb. Before this verb at least one element must be placed. %1 must be a slot, this slot must be filled. %2 is marked `mslot` (multiple slot). A multiple slot can be filled with an arbitrary number of elements but can also be empty. An arbitrary number of elements can also follow after the verb. It can also be described that the subject must go in the first position; in this case %1 has to be added to the subject slot. Please note that this is a lexicalised grammar, i.e. all information is stored in the lexicon, no extra grammar rules are needed.

4.2 An Example With a Discontinuity

Our treatment of free word order has already been introduced in the previous Section 4.1. Now a more complicated example introduced in Section 2 *János azt hiszem hogy elfogadja a javaslatot. (János, I think, that (he) accepts the proposal.)* is used to show how discontinuities are handled. The dependency tree has already been shown in Figure 2. In Figure 4 a short and not complete grammar for the example sentence is given. Templates are left out for simplicity. In the grammar, continuity and discontinuity are marked using special features. A feature `cont` is used to specify whether a dependency may be realized only continuously ("+"), only discontinuously ("-"), or both (not specified); a second feature `cont-const` is used to specify whether dependents may be extracted from the constituent headed by a certain word[3]. Both features can be applied to lexical entries of words, to slots, or to positions in a position list. Specification of dependents, slots or positions as continuous will stop this process from taking place.

To parse the sentence for example 4, the parser first encounters the words *János* and *azt (that-acc)*. *Azt (that-acc)* contains an open slot for a subjunction and

[3]No example for `cont-const` is given.

```
Word "János" [
  number: sing;
  person: 3;
  gender: masc;
  case: nom;
  lexeme: János;
  special: propername;
  category: noun;]

Word "azt" [
  conj: slot [
     category: subjunction;];
  lexeme: az;
  category: defpronoun;
  case: acc;
  order: (i);]

Word "hiszem" [
  category: verb;
  sentence: declarative;
  dir-obj: %2 slot [
      case: acc;
      category: defpronoun;];
  lexeme: hisz;
  numerus: sing;
  person: 1;
  order: (%1 %2 i %3);]
%1 = oslot [];
%3 = oslot [];

Word "hogy" [
    category: subjunction;
    lexeme: hogy;
    prop: %1 slot [
        category: verb;
        cont: +;];
    order: (i %1);]

Word "elfogadja" [
    category: verb;
    lexeme: elfogad;
    subj: oslot [category: noun;
                 case: nom;];
    dir-obj:  slot  [category: noun;
                     case: acc];
    order: (%1 i %2);]

Word "a" [
    category: determiner;
    lexeme: a;]

Word "javaslatot" [
    category: noun;
    case: acc;
    lexeme: javaslat;
    spec: %1 oslot [
        category: determiner;
        cont: +;];
    order: (%1 i);]
```

Figure 4: Simple grammar for *János azt hiszem hogy elfogadja a javaslatot. (János, I think, that (he) accepts the proposal.)*

therefore cannot act as head for *János*. The next word *hiszem (I think)* has an open slot for an definite pronoun in accusative. This slot can be filled with *azt (that-acc)*. This slot is marked with %2 and in the feature `order` of *hiszem (I think)* it is indicated, that it has to be positioned in front of the verb itself. *János* can fill position %1 in this list. %1 is not bound to any slot of the verb *hiszem (I think)* so the syntactic head of *János* is not available yet. The open slot of *azt (that-acc)* can not be filled with any word between *azt (that-acc)* and *hiszem (I think)*, so this slot is passed up to the syntactic head of *azt (that-acc)* which is *hiszem (I think)*. After *hiszem (I think)*, *hogy (that)* is found by the parser. *hogy (that)* can fill the slot of *azt (that-acc)* that has been passed up to *hiszem (I think)*. Additionally in the feature `order` of *hiszem (I think)*, *hogy (that)* can fill the position %3. *hogy (that)* has an open slot for an subordinate clause with a verb as head. `cont:+` indicates that this slot cannot be moved up. The next word word *elfogadja (accepts)* fills this slot and opens two new ones. One slot is for the subject, that is optional in Hungarian, he second slot is for an object. This object slot is filled by the final two words *a javaslatot (the proposal). javaslatot (proposal)* opens a slot for a determiner. This slot can not be moved up and the determiner must

be positioned in front of *javaslatot (proposal)*. *javaslatot (proposal)* then fills the object slot of *elfogadja (accepts)*. The subject slot of *elfogadja (accepts)* remains unfilled up to now. Because this slot can be discontinuous, it is now moved up the dependency tree until it reaches *hiszem (I think)*, where it can be filled with *János*. Similarly the sentences *Azt hiszem hogy János elfogadja a javaslatot*, *A javaslatot azt hiszem hogy János elfogadja.*, ... can be parsed with the help of this grammar. They have a similar meaning than the original sentence but stress different parts.

5 Algorithmic Description

The first subsection describes the simpler variant of the algorithm, which cannot deal with discontinuities [4], [16]. The subsequent subsections describe the extensions necessary to deal with discontinuities.

5.1 Parsing Sentences without Discontinuities

In the description of the algorithm, lexical entries without the complete feature structures will be used. The words' lexical entries are based on the part-of-speech K of a word and three lists:

- First for each word an unsorted list of empty slots $\mathcal{L}$, that can be filled, is necessary.
- Additionally, each lexical entry contains a position list P, that is split in two parts:
  - $P_{\leftarrow}$ contains the positions to the left side of the word described and
  - $P_{\rightarrow}$ contains positions to its right side.

  Both lists start with the position that is closest to the current word, the following entries are sorted according to the distance from the word described. Positions are named according to their entry in $\mathcal{L}$ with their part of speech from the unsorted list of empty slots or x, if they have no corresponding empty slot. Those x labelled entries can be filled by any word.

The basic data structure of the algorithm is a chart. When parsing a sentence $w_0 w_1 \ldots w_{n-1}$ consisting of n words, the chart is an $(n+1) \times (n+1)$ table. In this table, sets of chart entries are stored. Entries are never removed from the chart and are immutable after creation. A chart entry at the position (i, j) contains information about the partial derivation of the part of the sentence ranging from the word w_i to the word w_{j-1}. A chart entry is also referred to as chart edge. A chart edge is inactive if the lists $\mathcal{L}$, $P_{\leftarrow}$ and $P_{\rightarrow}$ are empty or contain only multiple and optional slots, written as $\langle\rangle$. Otherwise an entry is active, because it still has positions which have to be filled. Based on the chart, the parsing Algorithm 5.1 is used.

Algorithm 5.1 *main loop (continuous)*

1: **for** $j := 1$ to n **do**
2: **for all** lexical entries $(K_j, \mathcal{L}_j, P_{\leftarrow j}, P_{\rightarrow j})$ of a word a_j **do**
3: Insert $(K_j, \mathcal{L}_j, P_{\leftarrow j}, P_{\rightarrow j})$ in $m_{j-1,j}$
4: As long as changes are possible:
5: **for all** $i, k < j$ **do**
6: **if** $k_1 = (D, \langle\rangle, \langle\rangle, \langle\rangle) \in m_{i,k} \wedge k_2 = (R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}) \in m_{k,j} \wedge P_{\leftarrow} \neq \langle\rangle$ **then**
7: *extend*(k_1, k_2, i, j)
8: **end if**
9: **if** $k_1 = (R, \mathcal{L}_R, \langle\rangle, P_{\rightarrow}) \in m_{i,k} \wedge k_2 = (D, \langle\rangle, \langle\rangle, \langle\rangle) \in m_{k,j} \wedge P_{\rightarrow} \neq \langle\rangle$ **then**
10: *extend*(k_2, k_1, i, j)
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **if** $m_{0,n}$ contains an inactive edge **then**
16: **return** true
17: **else**
18: **return** false
19: **end if**

$m_{i,j}$ is the set of chart entries at (i, j). At the beginning all sets are empty. An active chart entry $(R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow})$, which serves as a head, can be extended with an inactive chart entry $(D, \langle\rangle, \langle\rangle, \langle\rangle)$, which serves as a dependent, if one entry is contained in $m_{i,k}$ and one entry is contained in $m_{k,j}$. The constituents described by these chart entries have to lie next to each other in the sentence which is parsed. The extension of chart edges is described in the Procedure 5.2 *extend*.

If the extension of a chart edge from $m_{i,k}$ and a chart edge $m_{k,j}$ was successful, the new edge is inserted in $m_{i,j}$ in the chart, as it contains a derivation for the words $w_i \ldots w_{j-1}$. It is only inserted into the chart if it is not contained in the chart yet. Therefore each entry is contained in the chart only once. If $P_{\leftarrow}$, the list containing the open slots left of the head is empty, i.e. all positions left of the head are filled, the algorithm tries to fill the first position on the right side of the head. A position can be filled if the parts of speech of the head and the position match.[4] For an example of a sentence with no discontinuities, which is parsed with this algorithm, see [16].

[4] And the corresponding feature structures can be unified in the parser.

Procedure 5.2 *extend*(k_1, k_2, i, j) *(continuous)*

Require: $k_1 = (D, \langle\rangle, \langle\rangle, \langle\rangle)$
Require: $k_2 = (R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow})$
1: **if** $P_{\leftarrow} = \langle\rangle$ **then**
2: $p := head(P_{\rightarrow})$
3: $P_{\rightarrow} := tail(P_{\rightarrow})$
4: **else**
5: $p := head(P_{\leftarrow})$
6: $P_{\leftarrow} := tail(P_{\leftarrow})$
7: **end if**
8: **if** $p = x$ **then**
9: **for all** $(d, X) \in \mathcal{L}_R$ with $X = D$ **do**
10: write $(R, \mathcal{L}_R - (d, X), P_{\leftarrow}, P_{\rightarrow})$ in $m_{i,j}$
11: **end for**
12: **else**
13: **for all** $(d, X) \in \mathcal{L}_R$ with $d = p \wedge X = D$ **do**
14: write $(R, \mathcal{L}_R - (d, X), P_{\leftarrow}, P_{\rightarrow})$ in $m_{i,j}$
15: **end for**
16: **end if**

5.2 Parsing Sentences with Discontinuities

The extension of Algorithm 5.1 to be able to handle discontinuities relies on the following observation. A discontinuous dependency is always part of a continuous constituent, as was described in Section 2. When a discontinuous dependency is established, it cannot be established a single step. It must be estalished in two steps, which might be separated by an arbitrary numbers of steps, because Algorithm 5.7 only tries to combine constituents which lie next to each other. The continuous constituent which contains both the dependent and head of the discontinuity and furthermore contains the position which the dependent fills was called head. A discontinuity is a relation between three words, a dependent, a syntactic head, which contains the slot for the dependent, and a linear head, which contains the position for the dependent. When encountering a discontinuity, two cases must be distinguished. Either the dependent or the syntactic head of the discontinuity appear first in the sentence. For an example where the dependent appears before the syntactic head take a look at the example sentence from Section 2, Figure 2. The dependent *János* is attached discontinuously to its syntactic head *elfogadja (accepts)* and *János* appears before *elfogadja (accepts)* in the sentence. The linear head of the discontinuity of *János* and *elfogadja (accepts)* is *hiszem (I think)*, because *hiszem (I think)* contains the position which the word *János* fills.

The following extensions were made to handle parsing with discontinuous constituents. A word is not a quadruple $(K, \mathcal{L}, P_{\leftarrow}, P_{\rightarrow})$ as in the continuous case, but becomes a quintuple $(K, \mathcal{L}, P_{\leftarrow}, P_{\rightarrow}, T)$, because words can be temporarily attached to other words. Temporarily attached words are stored in the list T and

are a mechanism to enable the parsing of discontinuities. Temporarily attached words are used to handle a discontinuity of the type where the parser encounters a dependent before its syntactic head. As the discontinuous variant of the parsing algorithm only attempts to unify constituents which are adjacent in the sentence, a dependent cannot fill a slot of its syntactic head directly. Therefore the dependent is attached temporarily to the linear head, until the slot from the syntactic head, which the dependent is supposed to fill, is moved up to the linear head. Then the dependent fills the slot. The slot from the syntactic head might be moved up several times before it reaches the linear head. When the dependent is attached to its linear head temporarily, it fills a position. This position must be a free position, which means it cannot be connected to a slot. Were the position connected to a slot, the dependent would fill a position and a slot, and would therefore be attached continuously.

Another mechanism mentioned in the paragraph above dealing with discontinuities is moving up slots. This extension deals with a discontinuity of the type where the syntactic head appears in the sentence before the dependent, as well as with the type of discontinuity mentioned in the above paragraph. As the syntactic head and the dependent cannot be combined directly, the slot which the dependent eventually fills is moved up to the feature structure of the head of the syntactic head when the syntactic head acts as a dependent and fills the slot of another constituent. This slot may be moved up several times, until it finally reaches the linear head. Then the dependent can fill the slot which was moved up.

If a chart entry has an open slot that can be realized discontinuously, this entry can be used as a dependent. This differs from the continuous algorithm. When parsing with continuous constituents, all slots must be filled before a chart entry can be used as a dependent. As mentioned before, discontinuous open slots can be moved up to be filled later. Therefore the Procedure 5.3 *extend* must be changed. To deal with linear heads and the position x (a position which is not connected to a slot) a new Procedure 5.4 *extend_free* is introduced. Words that fill a position from the position list of a linear head, are not unified with a slot $\mathcal{L}$ as in the continuous case. They are attached to a possibly linear head temporarily. Later the temporarily attached words will eventually have to fill a slot. What happens in that case is described in the Procedure 5.5 *reduce*. If a word or a constituent fills a position $p \neq x$ from $\mathcal{L}$, it is treated the same way as if it were attached to a word which cannot be a linear head.

5.2.1 Moving Up Slots and Attaching Words Temporarily

The Procedure 5.3 *extend* must be changed to move up slots and attach words temporarily. If this procedure is called, the possible dependent contains no temporarily attached words and both its position lists are empty. If the next position to be filled is connected to a slot, then the possible dependent is unified with the slot. A successful unification results in a new chart entry. If the next position to be filled is a free position, then the Procedure 5.4 *extend_free* deals with the dependent. The Procedure 5.4 *extend_free* checks whether the head can be a linear head. If that is

Procedure 5.3 *extend*(k_1, k_2, i, j) *(discontinuous)*

Require: $k_1 = (D, \mathcal{L}_D, \langle\rangle, \langle\rangle, \langle\rangle)$
Require: $k_2 = (R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}, T_R)$
if $P_{\leftarrow} = \langle\rangle$ **then**
  $p := head(P_{\rightarrow})$
  $P_{\rightarrow} := tail(P_{\rightarrow})$
else
  $p := head(P_{\leftarrow})$
  $P_{\leftarrow} := tail(P_{\leftarrow})$
end if
if $p = x$ **then**
  extend_free$(D, \mathcal{L}_D, R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}, T_R, i, j)$
else
  for all $(d, X) \in \mathcal{L}_R$ with $d = p \wedge X = D$ **do**
    if not ($\mathcal{L}_D \neq \langle\rangle \wedge (d, X)$ must be a continuous constituent) **then**
      insert $(R, \mathcal{L}_R - (d, X) + \mathcal{L}_D, P_{\leftarrow}, P_{\rightarrow}, T_R)$ in $m_{i,j}$
    end if
  end for
end if

the case the dependent is temporarily attached to the possibly linear head and a new chart entry is created. Otherwise the head and the dependent are treated the same as in Procedure 5.3 *extend*.

5.2.2 Filling Open Slots with Temporarily Attached Words

Another extension necessary to be able to handle discontinuities, is to deal with temporarily attached words. At some point during the parse those temporarily attached words eventually have to fill slots in the word w they have been attached to. Because this is an expensive operation, it is delayed as long as possible. For this reason, filling the remaining slots with temporarily attached words is only done, if the word's positions lists are empty and the word itself could possibly be attached to another word as a dependent. For every open slot the algorithm

Procedure 5.4 *extend_free* $(D, \mathcal{L}_D, R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}, T_R, i, j)$ *(discontinuous)*

if head can be a linear head **then**
  insert $(R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}, T_R + (D, \mathcal{L}_D))$ in $m_{i,j}$
else
  for all $(d, X) \in \mathcal{L}_R$ with $X = D$ **do**
    if not($\mathcal{L}_D \neq \langle\rangle \wedge (d, X)$ must be a continuous constituent) **then**
      insert $(R, \mathcal{L}_R - (d, X) + \mathcal{L}_D, P_{\leftarrow}, P_{\rightarrow}, T_R)$ in $m_{i,j}$
    end if
  end for
end if

Procedure 5.5 *reduce*(k, i, j) *(discontinuous)*

```
1: if T = ⟨⟩ then
2:   insert (R, L, ⟨⟩, ⟨⟩, ⟨⟩) in m_{i,j}
3: else
4:   if L ≠ ⟨⟩ then
5:     for all (d, X) ∈ L do
6:       for all (D, L_D) ∈ T with X = D do
7:         if not(L_D ≠ ⟨⟩ ∧ (d, X) must be a continuous constituent) then
8:           insert (R, L − (d, X) + L_D, ⟨⟩, ⟨⟩, T − (D, L_D)) in m_{i,j}
9:         end if
10:      end for
11:    end for
12:  end if
13: end if
```

attempts to unify every temporarily attached word with the open slot. For every successful unification, the result is stored in the chart. It should be noted that a temporarily attached word need not necessarily be attached discontinuously, that depends solely on the slot it fills. If the temporarily attached word fills a slot which was moved up, it is attached discontinuously. Otherwise it is attached continuously, see Procedure 5.5.

5.2.3 Checking for Possible Dependents

Another necessary extension is a method for determining whether a word is a possible dependent, the Procedure 5.6 *check_dep*. As before, all words which do not contain any slots can become a dependent. But in contrast to the continuous

Procedure 5.6 *check_dep*(k) *(discontinuous)*

```
Require: k = (D, L_D, ⟨⟩, ⟨⟩, ⟨⟩)
1: if L_D = ⟨⟩ then
2:   return true
3: else
4:   if L_D has continuous empty slots then
5:     return false
6:   else
7:     if Lexical entry of k must be continuous then
8:       return false
9:     else
10:      return true
11:    end if
12:  end if
13: end if
```

case, a word with open slots can become a dependent, if the open slots can be filled discontinuously and the word does not have to be a continuous constituent.

5.2.4 Main Parsing Algorithm

The main loop (Procedure 5.7) incorporates all the extensions made so far. Four different cases are considered in the main loop. The first and the second case are similar to those in the continuous main loop. One difference is that more words may be possible dependents, and open slots in a dependent are moved up when the dependent is attached to its head. A dependent may also be attached to its head temporarily. In the first case the dependent is to the left of the head, in

Algorithm 5.7 *main loop (discontinuous)*

1: **for** $j := 1$ to n **do**
2: **for all** lexical entries $(K_j, \mathcal{L}_j, P_{\leftarrow j}, P_{\rightarrow j})$ of a word a_j **do**
3: Insert $(K_j, \mathcal{L}_j, P_{\leftarrow j}, P_{\rightarrow j})$ in $m_{j-1,j}$
4: **for all** $i, k < j$ **do**
5: **if** $k_1 = (D, \mathcal{L}_D, \langle\rangle, \langle\rangle, \langle\rangle) \in m_{i,k} \wedge$ *check_dep*$(k_1) \wedge k_2 = (R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}, T_R) \in m_{k,j} \wedge P_{\leftarrow} \neq \langle\rangle$ **then**
6: *extend*(k_1, k_2, i, j)
7: **end if**
8: **if** $k_1 = (R, \mathcal{L}_R, \langle\rangle, P_{\rightarrow}, T_R) \in m_{i,k} \wedge k_2 = (D, \mathcal{L}_D, \langle\rangle, \langle\rangle, \langle\rangle) \in m_{k,j} \wedge$ *check_dep*$(k_2) \wedge P_{\rightarrow} \neq \langle\rangle$ **then**
9: *extend*(k_2, k_1, i, j)
10: **end if**
11: **if** $k_1 = (D, \mathcal{L}_D, \langle\rangle, \langle\rangle, T_D) \in m_{i,k} \wedge T_D \neq \langle\rangle \wedge k_2 = (R, \mathcal{L}_R, P_{\leftarrow}, P_{\rightarrow}, T_R) \in m_{k,j} \wedge P_{\leftarrow} \neq \langle\rangle$ **then**
12: *reduce*(k_1, i, k)
13: **end if**
14: **if** $k_1 = (R, \mathcal{L}_R, \langle\rangle, P_{\rightarrow}, T_R) \in m_{i,k} \wedge k_2 = (D, \mathcal{L}_D, \langle\rangle, \langle\rangle, T_D) \in m_{k,j} \wedge T_D \neq \langle\rangle \wedge P_{\rightarrow} \neq \langle\rangle$ **then**
15: *reduce*(k_2, k, j)
16: **end if**
17: **end for**
18: **end for**
19: **end for**
20: **for all** $k \in m_{0,n}$ with empty position list and non-empty list T **do**
21: *reduce*$(k, 0, n)$
22: **end for**
23: **if** $m_{0,n}$ contains an inactive edge **then**
24: **return** true
25: **else**
26: **return** false
27: **end if**

the second case to the right. If temporarily attached words are removed from the words they are attached to, the third and the fourth case come into play. Words are removed from the list T of a word w only if w might be a dependent. Finally temporarily attached words must be removed from chart entries $m_{0,n}$, which span the whole sentence. Otherwise possible solutions may not be found.

6 Conclusion and Future Work

Word order phenomena and discontinuity in Hungarian were modelled for a dependency parser. It turned out that, as for the other languages tested, it was possible and easy to write down. Nevertheless there is still a lot of work to do. Up to now only special problems of Hungarian have been modeled as a proof of concept for the parser. Next a full-flexed Hungarian grammar should be developed.

It is most important to add a Hungarian morphology to the parser. Up to now, the parser works with a full form lexicon for Hungarian. This might be a solution for other languages, but it does definitely not work for Hungarian due to the high number of possible word endings.

Acknowledgements

The authors thank Szilvia Svada and Gabriella Kókai for explaining all the tricky details of the Hungarian grammar.

References

[1] Norbert Bröker, *Separating surface order and syntactic relations in a dependency grammar*, in Proceedings of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics, Montreal, 1998.

[2] Michael A. Covington, *Parsing discontinuous constituents in dependency grammar*, Computational Linguistics, 16(4):234-236, 1990.

[3] Ralph Debusmann, Denys Duchier and Geert-Jan Kruijff *Extensible Dependency Grammar: A New Methodology*, Recent Advances in Dependency Grammar, COLING 2004.

[4] Ricarda Dormeyer, *Syntaxanalyse auf der Basis der Dependenzgrammatik*, PhD Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2004.

[5] Norman M. Fraser, *Parsing and dependency grammar*, UCL Working Papers in Linguistics, 1:296-319, 1989.

[6] Peter Hellwig, *Chart parsing according to the slot and filler principle*, in Proceedings of the 12th International Conference on Computational Linguistics, pages 242-244, Budapest, 1988.

[7] Richard Hudson, *Word Grammar*, Blackwell, Oxford, 1984.

[8] Richard Hudson, *Towards a computer-testable word grammar of English*, UCL Working Papers in Linguistics, 1:321-338, 1989.

[9] Daniel Jurafsky and James H. Martin, *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, New Jersey, 2000.

[10] László Keresztes, *Hungaro Lingua: Praktische ungarische Grammatik*, Debreceni Nyári Egyetem, 1999.

[11] Katalin Kiss, *The Syntax of Hungarian*, Cambridge Syntax Guides, Cambridge University Press, 2002.

[12] Michael C. McCord, *Slot grammar. A system for simpler construction of practical natural language grammars*, in Rudi Studer, editor, Natural Language and Logic, pages 118-145. Springer, Berlin, Heidelberg, 1990.

[13] Ivan Sag, Thomas Wasow and Emily Bender: *Syntactic Theory. A Formal Introduction*, Second Edition, Stanford: Univ. of Chicago Press, 2000.

[14] Lucien Tesnière, Esquisse d'une syntaxe structurale, Klincksieck, Paris, 1953.

[15] Lucien Tesnière, Eléments de syntaxe structurale, Klincksieck, Paris, 1959.

[16] Thomas Tröger, *Ein Chartparser für natürliche Sprache auf der Grundlage der Dependenzgrammatik*, Master Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2003.

[17] Alexandra Pröll, *Eine Dependenzgrammatik für das Japanische*, Bachelor Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2004.

[18] Gábor Prószéky, Ilona Koutny and Balázs Wacha, *A dependency syntax of Hungarian*, in Dan Maxwell and Klaus Schubert, eds., Metataxis in Practice, pages 151-182. Foris Publications, Dordrecht, 1989.

[19] Markus Schulze, *Ein sprachunabhängiger Ansatz zur Entwicklung deklarativer, robuster LA-Grammatiken*, PhD Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2004.

Acta Cybernetica **17** (2006) 633–646.

Named Entity Recognition for Hungarian Using Various Machine Learning Algorithms

Richárd Farkas*, György Szarvas† and András Kocsor*,‡

Abstract

In this paper we introduce a statistical Named Entity recognizer (NER) system for the Hungarian language. We examined three methods for identifying and disambiguating proper nouns (Artificial Neural Network, Support Vector Machine, C4.5 Decision Tree), their combinations and the effects of dimensionality reduction as well. We used a segment of Szeged Corpus [5] for training and validation purposes, which consists of short business news articles collected from MTI (Hungarian News Agency, www.mti.hu). Our results were presented at the Second Conference on Hungarian Computational Linguistics [7]. Our system makes use of both language dependent features (describing the orthography of proper nouns in Hungarian) and other, language independent information such as capitalization. Since we avoided the inclusion of large gazetteers of pre-classified entities, the system remains portable across languages without requiring any major modification, as long as the few specialized orthographical and syntactic characteristics are collected for a new target language. The best performing model achieved an F measure accuracy of 91.95%.

Keywords: named entity recognition, statistical models, machine learning

1 Introduction

The identification and classification of proper nouns in plain text is of key importance in numerous natural language processing applications. Take, for instance, Information Extraction systems (IE), where proper names generally play roles holding important information about the text itself, and thus are targets for extraction, or Machine Translation, which has to handle proper nouns and other sort of words in a different way, due to the specific translation rules that apply to them.

*MTA-SZTE, Research Group on Artificial Intelligence, 6720 Szeged, Aradi Vértanuk tere 1., Hungary, E-mail: {rfarkas,kocsor}@inf.u-szeged.hu

†University of Szeged, Department of Informatics, 6720 Szeged, Árpád tér 2., Hungary, E-mail: szarvas@inf.u-szeged.hu

‡The author was supported by the János Bolyai fellowship of the Hungarian Academy of Sciences.

The task of categorizing proper names was introduced during the 90s as a part of the shared tasks in the Message Understanding Conferences. The goal of these conferences was the identification and classification of proper nouns (like person, organization, location names), and phrases describing dates, time intervals, measures, quantities and so on in texts collected from English newspaper articles.

As a part of the Computational Natural Language Learning conference in 2002 and 2003 [21] a shared task dealt with the development of such systems that were able to identify the more difficult classes defined in earlier approaches at MUCs in two different languages at the same time. These less obvious categories they focused on were person, organization and geographical location names, along with all proper nouns that not belong to these three classes, treated as miscellaneous proper names. The text collection again consisted of newspaper articles, in Spanish + Dutch and English + German, respectively. The best performing systems gave an accuracy of 85-89% F measure for English (the best known results for any language). Since we follow the task definition of the CONLL shared task series, where the NER was focused mainly on proper names, the expressions 'proper noun/name recognition' and 'named entity recognition' are used synonymously here.

Research and development efforts in the last few years have focused mainly on other languages [15], [23], or domains like biological scientific texts [8], or cross-language recognition [16]. Hungarian named entity recognition fits in this trend quite well, due to the special agglutinative property of the Hungarian language, which makes most natural language processing tasks quite difficult.

Machine learning methods have been applied to the NER problem with remarkable success, and a few of these systems have proved to be efficient in disambiguating NEs in more than one language at the same time without major modification. While only one learning system [3] and seven rule-based classifiers were submitted to the last MUC contest (in 1997), only statistical algorithms took part in the shared task of CoNLL-2003 where the most frequently applied techniques were the Maximum Entropy Model (ME) [6] and Hidden Markov Models (HMM) [13]. Surprisingly only one ME and one clear HMM model was presented on the COLING 2004 Post-Conference Workshop (JNLPBA) [12]. Here the Support Vector Machine (SVM) [14] - used in isolation or in combination with other models - was the most popular.

There are some results on NER for the Hungarian language as well. A model based on expert rules defined by linguist experts was developed and tested at the Hungarian Research Institute on Linguistics [10], which also served as a basis for the proper noun recognizer module of HumorESK, a syntactic parser for Hungarian developed by MorphoLogic Ltd. [19]. However, to our knowledge, no statistical models have yet been made for the Hungarian language.

As we mentioned above, a wide variety of learning algorithms have been applied to the NER task. Our goal in this paper is not only to examine some models that were used less frequently, but to focus on the aggregation of dimensionality reduction and classifier combination schemes as well. Since the combinations of statistical methods with diverse theoretical bases often lead to even more accurate models, as it's often good to tackle a problem from many angles, we employed three

algorithms of a very different nature. We decided to make use of the C4.5 decision tree learning algorithm [20], feedforward artificial neural network [2], support vector machine [22] classifiers and their combination. Finally we tested the applicability of statistical models for NER in a language that has several special properties, making many Natural Language Processing (NLP) tasks rather difficult.

In the next section we will discuss the Named Entity Recognition task, then in Section 3 we introduce the learning model constructed by us to solve it. In Section 4 the machine learning algorithms we applied are briefly described followed by the description of some meta-learning methods that combine the hypotheses produced by the learners into a joint and hopefully more accurate hypothesis. In the experiments section we discuss in detail our investigation on each model separately, the combined model and the results of feature selection as well. After, in Section 6, we summarize the main characteristics of the Named Entity Recognizer system we developed, discuss our results and offer some suggestions for future research.

2 Named Entity Recognition for the Hungarian Language

The identification of proper names can be regarded as a tagging problem where the aim is to assign the correct tag (label) for each token in a plain text. This classification determines whether the lexical unit in question is part of a proper noun phrase and if it is, which category it belongs to.

Here we follow the task definition used in the CONLL conferences to distinguish four classes of entities: person names, organization names, place/geographic names, miscellaneous entity names. The latter class includes all proper nouns that do not belong to any of the other three classes. This way a comparison of our results with those published for other languages is more pertinent and this categorization is straightforward for IE purposes, where the system will be applied [1]. Earlier approaches define additional classes of phrases describing measures, dates and so on. Such phrases are rarely proper nouns, they are simpler to identify and at the University of Szeged there is another application that handles such phrases [18], so this task will not be included.

This entity identification and class assignment is not straightforward in many cases and even a human annotator might need additional information about the context and some background knowledge to decide the appropriate class. Take, for instance, the names of business organizations that can frequently refer to the product itself they make (Audi, Nokia, etc.) and hence such terms can belong either to the *organization* or the *miscellaneous* classes. Occasionally it may happen that a phrase can fall into any one of the four categories, depending on the context (like "Ford", which can refer to a person, the concern, an airport or the car type).

In many approaches given in the literature the starting tokens of Named Entities are distinguished from the inner parts of the phrase [21]. This turns out to be useful when several proper nouns of the same type follow each other, because it enables the system to separate them instead of treating them as one entity. When doing so,

one of the "I-", "B-" (for inside and begin) labels also has to be assigned to each term that belongs to one of the four classes used. In our experiments we decided not to do this for two reasons. First, for some of the classes we barely have enough examples in our dataset to separate them well, and it would make the data available even more sparse. Second, in Hungarian texts, proper names following each other are almost always separated by punctuation marks or a stopword. We manually checked a small part (3%) of our training corpus and found only one case where such a distinction would have proved to be useful, so it definitely would have done more harm than good to the model to deal with this.

A segment consisting of short business news articles of Szeged Corpus [5] was used for training and testing the model. Our dataset consisted of 9600 sentences altogether, which had a full syntactic annotation[1], and the correct classification of NEs had also been added. In our model we only used the part-of-speech tags out of the several syntactic features, because a proper name classifier module should come immediately after the POS tagging in an information extraction application. This way the output of the NE tagger can aid the performance of the syntactic parser with the class labels assigned and the contraction of several tokens into one proper noun. Several articles in the literature claim that more knowledge about the syntax (identification of heads of noun phrases for example) can be beneficial to NE recognition [9], (in CONNL2003, 9 out of 16 systems used chunking information for English NER, including the best performing system [21]), but as we stated earlier this works both ways thus we decided not to use such information. This way recognition with our system can be performed without intensive syntactic analysis. To get part-of-speech information we used a morphologic analyzer (Humor), a guesser module that does rough a morphological analysis of words judged to be unknown words by Humor and a part of speech tagger (we did not use the disambiguated POS codes in the corpus to model a real life application).

The data we used also has some interesting properties regarding the distribution of class labels which arise from the domain specificity of the texts. The organization class for instance, which turned to be harder to recognize than person names, occurs more often in our corpus than those used in the CONLL conferences.

Table 1: Corpus details

	Tokens	Phrases
Non-tagged tokens	200067	-
Person names	1.921	982
Organizations	20.433	10.533
Locations	1.501	1.294
Misc. entities	2.041	1.662

[1]The project was carried out together with MorphoLogic Ltd. and the Academy's Research Institute for Linguistics

3 The learning model

To build a learning model we collected various types of numerically encodable information describing each term and its surroundings. These constituted the vector of attributes for the classification (which is partially based on the model described in [6]). A subset of the features used tries to capture the orthographical regularities of proper nouns, like capitalization, inner-word punctuation and so on. Another set of attributes denotes the role of the word and its neighboring words in the sentence; part of speech and case codes are examples of information of this type. The remaining parameters try to formalize and provide the system with some of the background knowledge a human annotator has and uses for disambiguating named entities. These are various lists of trigger words, ratios of the word appearing capitalized and lowercased in large corpora (or in everyday language).

The word form itself, along with gazetteers of pre-classified NEs frequently appearing in business news were not used to aid recognition, to improve the generalization capability of the system on unknown words and texts. Using such word lists would undoubtedly increase the performance on domain specific corpora, by reducing the need for the automatic classification to those words that cannot be found in the database. Practical natural language applications usually have their own, domain specific lexicons, and automatic named entity recognition can help to them when these lexicons do not cover the parsed text entirely. In these cases a general-purpose NE lexicon is likely to be of little use.

The features we employed were the following:

- part-of-speech code (for the word itself and for its +/-4 words neighborhood),
- case code,
- type of initial letter of the word,
- one that tells if the word contains digits inside the word form,
- one that tells if the word contains capitalized letter inside the word form,
- one that tells if the word contains punctuation inside the word form,
- the word in the beginning of a sentence or not,
- the word between quotation marks or not,
- word length,
- the word is an arabic or roman number,
- memory that tells whether the word in question received an NE tag earlier in the actual document (one article) or not, and what type,
- the ratio of lowercase and capitalized frequency in Szószablya [11] term frequency dictionary,
- the ratio of mid-sentence uppercase frequency and general uppercase frequency in Szószablya,

- feature telling us whether the word is in one of the trigger word dictionaries (we used dictionaries of surnames, organization forms, geographic name types and stopwords, and a very small gazetteer containing the names of the countries of the world and names of the biggest cities).

Using the above features - with unary representation, except for the word length - each term had 120 different attributes and the correct NE label assigned to it. This way Named Entity Recognition is defined as a classification problem which can be solved using algorithms which apply some kind of inductive learning approach.

To evaluate our system we used two reasonably well performing naive algorithms. The first one was based on the following decision rule: *For every term that is part of an entity assign the organization class.* This simple method achieved a score of 71.9% precision and 69.8% recall on the evaluation sets (70.8% F measure). These good results are due to the fact that the knowledge of what an NE is (and is not) was added to the baseline algorithm and the characteristics of the domain (in business news articles the *organization* class dominates the other three). The second baseline algorithm selected the complete unambiguous named entities appearing in the training data and attained a 77.81% F measure accuracy. These results are slightly better than those published for different languages, which is once again due to the unique characteristics of business news texts where the distribution of entities is biased towards the organization class.

4 Learning algorithms, meta-learning, attribute selection

To solve classification problems effectively it is worth applying various types of classification methods, both separately and in a combination. Since the optimal single or hybrid method varies from domain to domain, to perform NE recognition we decided to test methods from diverse areas of machine learning. Apart from the classifier selection problem, however, other issues have to be dealt with as well. For instance, using an insufficiently high dimensional embedding feature space for constructing the classification models may have certain drawbacks. Superfluous features can have a detrimental influence on the classification methods. A general observation is that performing a careful dimension reduction - prior to learning - can significantly increase the classification performance. This is true for NLP applications in particular, where features are often correlated and might hold little evidence on the target variable. In this section we describe the learning algorithms applied, the classifier combination technique used and the feature (or attribute) selection methodology we employed during the tests.

4.1 C4.5

C4.5 [20] is based on the well-known ID3 tree learning algorithm. It is able to learn pre-defined discrete classes from labeled examples. The result of the learning

process is an axis-parallel decision tree. This means that during the training, the sample space is divided into subspaces by hyperplanes that are parallel to every axis but one. In this way, we get many n-dimensional rectangular regions that are labeled with class labels and organized in a hierarchical way, which can then be encoded into the tree. Since C4.5 considers attribute vectors as points in an n-dimensional space, using continuous sample attributes naturally makes sense. For knowledge representation, C4.5 uses the "divide and conquer" technique, meaning that regions are split during learning whenever they are insufficiently homogeneous. Splitting is done by axis-parallel hyperplanes, and hence learning is very fast. One great advantage of the method is time complexity; in the worst case it is $O(dn^2)$, where d is the number of features and n is the number of samples. Based on this we used the C4.5 algorithm lots of times to perform preliminary tests to decide whether the inclusion of further features in beneficial to the model or not.

4.2 Artificial Neural Networks (ANN)

Since it was realized that, under proper conditions, ANNs can model the class posteriors [2], neural nets have become evermore popular in the Natural Language Processing community. ANNs are based on the parallel architecture of the brain. We can imagine it as a simple multiprocessor system with a large number of interconnections and interactions between the processing units using scalar messages. Describing the mathematical background of the ANNs is beyond the scope of this article. Besides, we believe that they are well known to those who are acquainted with pattern recognition. In the ANN experiments we used the most common feed-forward multilayer perceptron network with the backpropagation learning rule.

4.3 Support Vector Machines

Theoretical discoveries generally have their own very different, unique histories before they find any practical application. One such example is the "kernel-idea", which had appeared in several fields of mathematics and mathematical physics before it became a key notion in machine learning. The kernel idea can be applied in any case where the input of some algorithm consists of the pairwise dot (scalar) products of the elements of an n-dimensional dot product space. In this case, simply by a proper redefinition of the two-operand operation of the dot product, we can have an algorithm that will now be executed in a different dot product space, and is probably more suitable for solving the original problem. Of course, when replacing the operand, we have to satisfy certain criteria, as not every function is suitable for implicitly generating a dot product space. The family of Mercer Kernels is a good choice (according to Mercer's theorem) [17]. The well-known and widely used Support Vector Machines (SVMs) [22] is a kernel method that separates data points of different classes with the help of a hyperplane. The created separating hyperplane has a margin of maximal size with a proved optimal generalization capacity. Another significant feature of margin maximization is that the calculated

result is independent from the distribution of the sample points. Perhaps the success and the popularity of this method can be attributed to the above property.

4.4 Stacking

The learning methods discussed above showed promising results during the experiments we made, the best models achieving or even exceeding 90% accuracy rate. There are several well known meta-learning algorithms in the literature that can lead to a 'better' model (in terms of classification accuracy) than those serving as a basis for it, or can significantly decrease the time consumption of the learning phase without loss of accuracy. Since in our case the time needed for learning was not really relevant (all the three algorithms can deal with a database of this size in a relatively short time, ranging from several minutes to a few hours), we concentrated on improving the accuracy of the system. Decreasing the learning time by dividing the training data between different learners would be necessary for a corpus with a size of 1 million tokens or more.

To combine the results we used the Stacking method, which is in fact a decision function defined over several different learners trained on the same dataset. In many cases the forecast produced by this decision (or we can say voting) function achieves better accuracy rates than the initial models. The theoretical assumption underlying Stacking is that hypotheses made by inherently different learning methods usually cover different parts of the solution space well, which means that their error surface is not necessarily the same. This way hybrid methods can combine the good characteristics of the individual models, leading to a better overall performance.

The decision function we used to integrate the three learnt hypotheses was the following: *if any two of three learners' output coincide we accept it as a joint prediction, and use the forecast of the best performing model neural network otherwise (if three different answers are given by the three models).*

4.5 Attribute selection

In a learning problem the features used, their values constitute a vector space, and the goal is to separate individual points in this space that have different class labels. Our parameter space had 120 dimensions, each attribute having only integer values, the majority of them lying in the $\{0, 1\}$ set. The attributes rarely describe the target function equally well.

In inductive learning, we approximate the value of a target variable based on a set of features, making a hypothesis for the $x_0, x_1, ..., x_s \rightarrow y$ mapping based on a number of pre-defined examples with known attributes and target values. In theory, we can assume that irrelevant attributes are not selected via the learning process, and will have no impact on our hypothesis. In practice this is not always the case: bad features can detrimentally affect system performance and of course increase the time needed to set the hypothesis (learning time).

5 Experiments

Evaluation set. To test the model we used an annotated corpus of 200,000 words, divided into 96 XML files. We set up 10 test cases, randomly choosing 82 files for training, 5 files for the development phase testing and 9 files for evaluation purposes. The results presented were calculated using these 10 test cases as a 10-fold cross validation.

Results. The accuracy scores of the three learning methods are quite close to each other, with the artificial neural network slightly outperforming the other two. We should note though that with some proper search (via grid search, a genetic algorithm, or simulated annealing for example) in the parameter space of the learners would possibly weaken this superiority of neural networks. We used the C4.5 decision tree learner with a confidence factor of 0.33 for pruning, the ANN with one hidden layer of one and half the number of hidden units than input neurons, using sigmoid activation functions, 50 epochs of training with a 0.3 learning rate. For SVM classification we had a cosine polynomial kernel of 3rd degree, with 3000 data points used as basis in the optimization. The three models attained an average performance of 90.79%, 91.24%, and 90.66% with a standard deviation of 2.52%, 2.09% and 2.43% respectively on the 10 test cases. Although the support vector classifier achieved a slightly worse performance than the other two methods, it had outstanding precision measure on the three less frequent classes and this was especially helpful for a voting model.

Voting. Since ANN was clearly superior to the other two learners (it had very competitive performance on most of the 10 test cases, and had the best F value on the development phase test sets as well with 91.66% against 91.4% and 90.85%) we used the following decision function as a voting rule between the learners: *if any two of three learners' output coincide we accept it as a joint prediction, and use the forecast of the best performing model neural network otherwise (if three different answers are given by the three models).*

Such a decision function can have one of three possible effects on the system output: it may change a bad answer of the ANN into a good one (make a correction), or it may change a correct forecast of the network into a bad one predicted by the other two learners; and finally it can have no effect on the outcome when a bad answer is changed to another bad one. Voting brings a gain in accuracy if the corrections occur more often than messing up the prediction - in our case the ratio of good/bad revisions was 64.06% good to 35.94% bad. The following table shows the different types of corrections on each test cases:

This voting scheme performed better than the initial learners, with an F measure of 91.95% on average. This is a slight improvement (0.71%) compared to ANN, but this small increase in accuracy meant a 8,11% decrease of error rate relatively to the neural network, which is significant.

The results presented in Table 3 averages the individual results of the 10 test cases weighted by the number of NEs in the evaluation sets. We did this because we separated train/test/evaluation sets based on the XML file units of the corpus. This way the number of NEs differed slightly form one test case to another, but

Table 2: Voting performance (on all 10 test cases).

Testcase	0	1	2	3	4	5	6	7	8	9	SUM
Revisions	100	86	99	103	65	71	91	68	76	101	860
Good	74	45	53	47	44	39	54	30	54	59	499
Bad	18	36	31	37	16	27	31	28	18	38	280
Irrelevant	8	5	15	19	5	5	6	10	4	4	81

they preserve the topical integrity of the corpus, with whole newspaper articles falling into one set without any splitting. With this weighting our measures are evenly balanced.

Table 3: Results of various learning models and the voting model (averages on 10 testcases).

	ANN	C4.5	SVM	Voting
	Precision / Recall / F measure (%)			
Location	83.4/69.4/75.8	83.2/70.9/76.6	92.9/33.2/48.7	90.8/68.0/77.7
Organization	93.1/95.9/94.5	92.5/95.8/94.2	90.6/97.9/ 94.1	92.1/97.8/94.9
Person names	84.1/82.8/83.4	79.7/79.7/79.7	86.7/74.9/ 81.1	87.1/80.9/83.9
Miscellaneous	82.2/60.7/69.8	80.9/60.9/69.5	95.0/54.1/ 69.4	91.4/56.9/70.1
OVERALL	**91.8/90.7/91.2**	**91.1/90.5/90.8**	**90.7/90.6/ 90.7**	**92.2/91.7/92.0**
IMPROVEMENT TO THE BEST BASELINE	**9.1/25.3/17.3**	**8.3/25.0/16.7**	**7.8/25.2/ 16.5**	**9.7/26.6/18.2**

Attribute selection. We used the statistical chi-squared test to rank the 120 attributes we had and we examined the system performance in the function of the number of features used. We chose one of the testcases to analyze the effect of discarding some of the attributes that was the most similar to the overall results obtained in our experiments. By doing this we hoped to obtain information relevant to all 10 problem sets. The effects of feature space size on system performance using C4.5 decision tree learner for classification can be seen in Figure 1. The dotted line represents the accuracy on the development phase test set and the continuous line denotes the F measure value on the evaluation set. As can be seen, the minimal representative feature set size is somewhere around 30 features, while 60 is the optimal space dimension for the decision tree classifier. This is an interesting and useful result.

Using the chi-squared test to rank attributes the capitalization information for a +/-1 interval, frequency ratio features from the Szószablya corpus, the "contains digit" attribute, POS + case code, some trigger word list features (surnames, geographic name and company types) and word length proved to be the most significant. Performing this kind of analysis using the other two models, ANN and SVM according to the optimal feature set size would be more time consuming, as

decision tree learning is much faster than the numeric learners, and exceeds the limitations of this article. The results obtained with an ideal embedding feature space dimensionality for C4.5 shows that attribute selection certainly enhances system performance, so investigating the optimal amount of attributes for a neural network and support vector classifier is an area we plan to study in the near future.

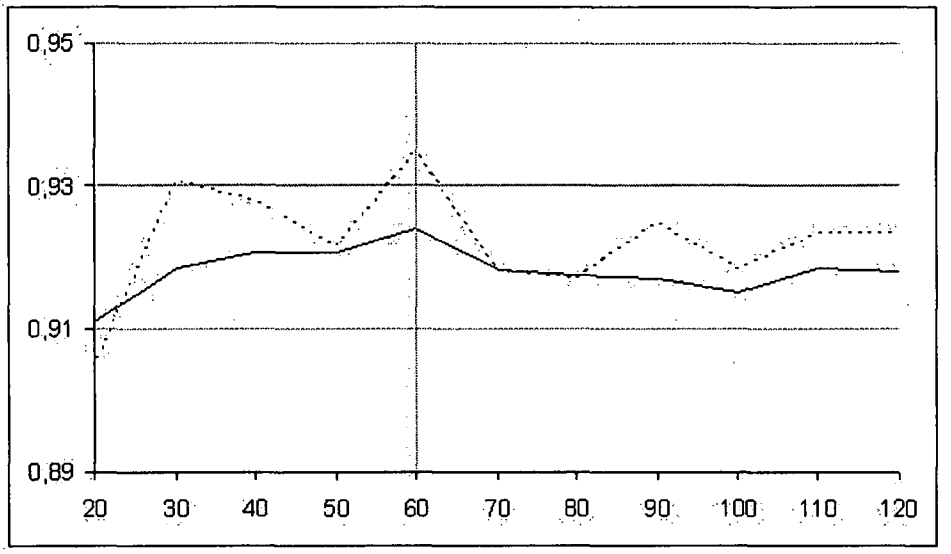


Figure 1: Effect of attribute selection on accuracy of C4.5 (number of attributes selected / F measure).

Results with attribute selection. After noting the results of Figure 1 we decided to keep only 60 features and then we redid all the experiments performed earlier. The decision tree learner increased its overall accuracy because we got rid of the many features that had little statistical relevance to the target class. C4.5 in the filtered feature space achieved a 91.38% F measure, which was better than any of the three individual models using all 120 attributes, yielding a 6.41% decrease in the relative error rate for the C4.5 classifier and 1.6% compared to the best individual model. ANN and SVM on the other hand produced poorer results this way (90.95% and 90.23% F values respectively) and it indicates that these numeric learners managed to capture some information from those less significant features that only confused the decision tree. A voting scheme that treated C4.5 as the superior model gave fair result (91.82% accuracy), showing that the loss of two models overcame the improvement of the third, but the system still benefited from voting.

We should note here however that optimal feature space size has not yet been investigated for numeric learners, so perhaps choosing a bigger dimension would be fruitful for the neural network or support vector classifier.

6 Summary and Conclusions

In this paper we presented an NE classifier system for the Hungarian language based on various statistic learning algorithms. Our main aim was to investigate the named entity recognition problem for Hungarian, which is quite a special language regarding its grammatical characteristics (e.g. its agglutinative nature). We tested the performance of artificial neural network, the C4.5 decision tree and support vector machine models on a classification problem using five different standard target classes used in recent conferences (CoNLL 2002, 2003) focused on named entity recognition for other languages. We constructed a parameter space of 120 dimensions, capturing orthographic, syntactic features of the text and background knowledge useful for disambiguating entity names.

Here we sought to demonstrate that statistical models can compete with expert rule-based systems that exist for Hungarian, and to discover whether feature selection and some meta-learning model can enhance overall system performance or not.

The target domain we chose were texts of newspaper articles, using a segment of the syntactically annotated Szeged Corpus consisting of business news collected from the homepage of the Hungarian News Agency (MTI, www.mti.hu). Business news articles are used as input data for other researches at the University of Szeged; we also plan to integrate our model into an information extraction system developed by researchers of the University and investigate its effects.

Our results show that statistical models can successfully disambiguate proper names and, unlike other natural language processing tasks like syntactic parsing, NE recognition can achieve competitive results in the Hungarian language to similar models for different languages or expert rule-based systems for Hungarian. Our best performing system achieved an accuracy of 91.95% in F measure, one that is close to the best results given in the literature, and competes with other NE classifiers developed for Hungarian. However, a cross-language comparison is quite risky and its relevance is questionable, while systems for Hungarian have been evaluated on a different corpus.

The experiments conducted here also show that meta-learning schemes and attribute selection can contribute to NE recognition, we got a significant improvement in the F measure using a voting rule (0.71% higher F value, 8.11% decrease of error rate, relative to ANN) and by using chi-squared test to rerank and filter the attributes that constitute the feature space for learning (a 0.59% increase in the F value, a 6.41% decrease of error rate of C4.5, relative to that with 120 attributes).

In the experiments we excluded some deep-knowledge features that could further increase classification accuracy, such as syntactic information about the text in order to build a system that can be used in practice and can be integrated into end-user applications like IE.

There are several other ways we might improve our system's performance on Hungarian business texts. These include using web search [4] to give further features according to word collocation with trigger words, using unannotated data, and using a Hidden Markov Model-based classifier which could be beneficial to the

model presented in our paper. Since system accuracy constantly increases with training set size, we expect that enlarging our training database with more manually annotated examples would also be helpful.

Currently we are testing our learning model on other languages like English to get more relevant comparisons to other systems.

References

[1] Alexin, Z., Csirik, J., and Gyimóthy, T. Programcsomag információkinyerési kutatások támogatására. In *Proceedings of II. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2004)*, pages 41–49, 2004.

[2] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., 1995.

[3] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[4] Cimiano, P. and Handschuh, S. Towards the self-annotating web. In *Proceedings of World Wide Web Conference 2004*, pages 462–471, 2004.

[5] Csendes, D., Csirik, J., and Gyimóthy, T. The szeged corpus: A pos tagged and syntactically annotated hungarian natural language corpus. In *Proceedings of the 7th International Conference on Text, Speech and Dialogue (TSD 2004)*, pages 41–47, 2004.

[6] Curran, J. R. and Clark, S. Language independent ner using a maximum entropy tagger. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 164–167. Edmonton, Canada, 2003.

[7] Farkas, R. and Szarvas, Gy. Statisztikai alapú tulajdonnév-felismero magyar nyelvre. In *Proceedings of II. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2004)*, pages 136–141, 2004.

[8] Finkel, J., Dingare, S., Nguyen, H., Nissim, M., Manning, C., and Sinclair, G. Exploiting context for biomedical entity recognition: From syntax to the web. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, 2004.

[9] Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. Named entity recognition through classifier combination. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada, 2003.

[10] Gábor, K., Héja, E., Mészáros, Á., and Sass, B. Nylt tokenosztályok reprezentációjának technológiája. Technical report, Academys Research Institute for Linguistics, Hungary, 2002. IKTA-00037/2002.

[11] Halácsy, P., Kornai, A., Németh, L., Rung, A., I., Szakadát, and V., Trón. A szószablya projekt www.szoszablya.hu. In *Proceedings of I. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2003)*, pages 298–299, 2003.

[12] Kim, J-D., Ohta, T., Tsuruoka, Y., and Y., Tateisi. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland*, 2004.

[13] Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. Named entity recognition with character-level models. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 180–183. Edmonton, Canada, 2003.

[14] Lee, C., Hou, W.-J., and Chen, H.-H. Annotating multiple types of biomedical entities: A single word classification approach. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, 2004.

[15] Màrquez, L., de Gispert, A., Carreras, X., and Padro, L. Low-cost named entity classification for catalan: Exploiting multilingual resources and unlabeled data. *Proceedings of Workshop on Multilingual and Mixed-language Named Entity Recognition, ACL 2003*, pages 25–32, 2003.

[16] Maynard, D., Tablan, V., and Cunningham, H. Ne recognition without training data on a language you don't speak. In *Proceedings of Workshop on Multilingual and Mixed-language Named Entity Recognition, ACL 2003*, 2003.

[17] Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, pages 415–446, 1909.

[18] Miháczi, A., Németh, L., and Rácz, M. Magyar szvegek természetes nyelvi elfeldolgozása. In *Proceedings of I. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2003)*, pages 38–44, 2003.

[19] Prószéky, G. Morphological analyzer as syntactic parser. In *Proceedings of 16th International Conference on Computational Linguistics (COLING-96)*, pages 1123–1126, 1996.

[20] Quinlan, J. R. *C4.5: Programs for machine learning.* Morgan Kaufmann, 1993.

[21] Tjong Kim Sang, E. F. and De Meulder, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[22] Vapnik, V. N. *Statistical Learning Theory.* John-Wiley & Sons Inc., 1998.

[23] Wu, Y., Zhao, J., and Xu, B. Chinese named entity recognition combining statistical model wih human knowledge. In *Proceedings of Workshop on Multilingual and Mixed-language Named Entity Recognition, ACL 2003*, 2003.

Acta Cybernetica 17 (2006) 647–659.

Learning Tree Patterns for Syntactic Parsing

András Hócza*

Abstract

This paper presents a method for parsing Hungarian texts using a machine learning approach. The method collects the initial grammar for a learner from an annotated corpus with the help of tree shapes. The PGS algorithm, an improved version of the RGLearn algorithm, was developed and applied to learning tree patterns with various phrase types described by regular expressions. The method also calculates the probability values of the learned tree patterns. The syntactic parser of learned grammar using the Viterbi algorithm performs a quick search for finding the most probable derivation of a sentence. The results were built into an information extraction pipeline.

1 Introduction

Syntactic parsing is the process of finding the immediate constituents of a sentence that is a sequence of words. Syntactic parsing is an important part of the field of natural language processing and it is useful for supporting a number of large-scale applications including information extraction, information retrieval, named entity identification, and a variety of text mining applications.

The Hungarian language is customarily defined as an agglutinative, free word order language with a rich morphology. These properties make its full analysis difficult compared to Indo-European languages. Unambiguous markers for the automatic rec-ognition of phrase boundaries do not exist. For example, the right bound of noun phrases could be the nouns as a head, but there is a possibility of omitting noun phrase head using its modifiers only. Determining the left bound of noun phrases is harder than the head, as it could be a determinant element. However, due to the possibility of a recursive insertion, it is not easy to decide which determiner and head belong to-gether. These difficulties mean that it is a quite hard to perform syntactic parsing with expert rules. In many cases the decision of annotators is based on semantic features rather than syntactic or structural ones.

An efficient solution for the problem of syntactic parsing might be the application of machine learning methods, but this requires a large number of training

*University of Szeged, Department of Informatics, H-6720 Szeged, Árpád tér 2., Hungary, E-mail: hocza@inf.u-szeged.hu

and test examples of annotated syntax trees. Since the Szeged Corpus[1] [3] became available, new methods have begun to be developed for syntactically parsing Hungarian sentences. The corpus contains texts from five different topic areas and currently has about 1.2 million word entries, 145 thousand different word forms, and an additional 225 thousand punctuation marks. The corpus contains manually POS tagged and disambiguated sentences and it was parsed by annotators who marked noun phrase structures and various clause structures.

After the completion of the annotation work the Szeged Corpus was then used for training and testing machine learning algorithms to retrieve syntactic grammars. For this the PGS (Pattern Generalization and Specialization) algorithm is introduced here, an improved version of the RGLearn algorithm [6] for learning syntactic tree patterns. The initial grammar for learner is collected from the Szeged Corpus using tree shapes to disassemble syntactic trees. Probability values of learned tree patterns are also computed. Based on this probability grammar the parsing algorithm incorporated in the Viterbi search method [18] can find quickly the most probable derivation of a sentence.

This paper is organized as follows. In Section 2 there is a review of related works and a description of efforts made by Hungarian researchers. Section 3 introduces a method used for collecting initial grammar for a learner from a large annotated corpus. Section 4 then presents the method used for learning grammar from an annotated corpus and extending this grammar with probability values using statistics. Section 5 introduces our method of syntactic parsing. After, Section 6 presents the test results we obtained. Finally, conclusions and suggestions for future study are given in Section 7.

2 Related works

Several authors have published results of syntactic parsing mainly for English. Generally the performance is measured with three scores. First, with a percentage of detected noun phrases that are correct (precision). Second, with a percentage of noun phrases in the data that is found via the classifier (recall). And third, with the $F_{\beta=1}$ rate which is equal to 2*precision*recall/(precision*recall). The latter rate has been used as the target for optimization.

Abney [2] proposed an approach which starts by finding correlated chunks of words. Ramshaw and Marcus [12] built a chunker by applying transformation-based learning ($F_{\beta=1}$=92.0). They applied their method to two segments of the Penn Treebank [10] and these are still being used as benchmark data sets. Argamon [4] uses memory-based sequence learning ($F_{\beta=1}$=91.6) for recognizing both NP chunks and VP chunks. Tjong Kim Sang and Veenstra [15] introduced cascaded chunking ($F_{\beta=1}$=92.37). The novel approaches attain good accuracies using a system combination. Tjong Kim Sang [14] utilized a combination of five classifiers ($F_{\beta=1}$=93.26).

[1] Magyar Távirati Iroda, http://www.mti.hu

Up to now there has been no good-quality syntactic parser available for Hungarian. Benchmark data sets for correctly comparing results on Hungarian do not exist yet either. The HumorESK syntactic parser [8] developed by MorphoLogic Ltd uses attribute grammar, assigning feature structures to symbols. The grammar part employed in the parser was made by linguistic experts. Another report on the ongoing work of a Hungarian noun phrase recognition parser [17] is based on the idea of Abney [1] using a cascaded regular grammar. The input to the grammar was a morpho-syntactically annotated text using a scaled-down version of the annotation scheme developed for the Hungarian National Corpus [16]. The grammar was developed by linguistic experts with the help of the CLaRK corpus development system [7] and it was tested in a short text of annotated sentences ($F_{\beta=1}$=58.78). The idea of using cascaded grammars seems beneficial, this technique being used in all Hungarian parser developments. A noun phrase recognition parser [6] applied machine learning methods to produce a grammar of noun phrase tree patterns from an annotated corpus ($F_{\beta=1}$=83.11).

3 Producing initial grammar for machine learning

Preprocessing provide training and test examples for machine learning. This process collects examples from the annotated corpus and transforms information into that needed for the learning problem. According to our approach the collected examples are tree patterns that can be used as grammar in a syntactic parser to help rebuild syntax tree of sentences. The role of machine learning is the generalization of this grammar to achieve good accuracy scores on unknown sentences as well.

3.1 Data source for training and evaluation

In order to perform well and learn from the various *Natural Language Processing* (NLP) tasks and achieve a sufficient standard of *Information Extraction* (IE), an adequately large corpus had to be collected that serves as the training database. While setting up various NLP projects, a relatively large corpus of varous types of texts was collected: the Szeged Corpus [3]. For demonstration purposes in the above-mentioned projects the authors chose a collection of short business news items issued by the Hungarian News Agency[2]. The chosen 6453 articles form part of the Szeged Corpus and relate to companies, financial and business life. The Szeged Corpus contains 1.2 million words, 225 thousand punctuation marks, and comes in an XML format using the TEIXLite DTD. The first version of the corpus contains texts from five topic areas, roughly 200 thousand words each, meaning a text database of some 1 million words. The second version was extended with a sample of texts of business news totalling another 200 thousand words. The texts are divided into sections, paragraphs, sentences and word entries.

Initially, corpus words were morpho-syntactically analysed with the help of the Humor [11] automatic pre-processor and then manually POS tagged by linguistic

[2]MTI, Magyar Távirati Iroda (http://www.mti.hu), "Eco" service.

experts. The Hungarian version of the internationally acknowledged MSD (Morpho-Syntactic Description) scheme [5] was used for encoding words. Due to the fact that the MSD encoding scheme is extremely detailed (one label can store morphological information on up to 17 positions), there are a lots of ambiguous cases, i.e. roughly every second word of the corpus is ambiguous. Disambiguation therefore required accurate and detailed work. Actually 64 person-months of manual annotation was needed for this. Currently all possible labels as well as the selected ones are stored in the corpus. About 1800 different MSD labels are used in the annotated corpus. The MSD label corresponds to the part-of-speech determined attribute, and specific characters in each position indicate the value for that attribute.

For example, the MSD label Nc-pa can be understood as

POS:	Noun,
Type:	common,
Gender:	- (not applicable to Hungarian),
Number:	plural,
Case:	accusative.

All the texts of Szeged Corpus were parsed manually, that is annotators marked various phrase structures. The extensive and accurate manual annotation of the texts, which required 124 person-months of manual work, is a good feature of the corpus. The syntax trees of annotated sentences contain various type of phrases, shown by following list:

Noun phrase (NP)	Verb prefix (PREVERB)
Adjective phrase (ADJP)	Conjunction (C)
Adverb phrase (ADVP)	Pronoun phrase (PP)
Verb phrase (VP)	Clause (CP)
Infinitive(INF)	Sentence (S)
Negative (NEG)	

3.2 Converting syntax trees to set of rules

Figure 1 shows a one-level pattern retrieving process. The tree disassembly is executed bottom-up, step by step, where a step consists of following substeps:

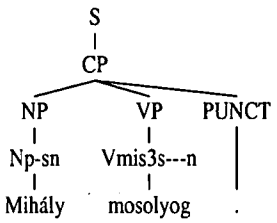
1. The exploration of base phrases (innermost phrases).
2. The storing base phrases.
3. Substituting base phrases with appropriate terminal symbols.

The tree disassembly process is completed when only the topmost symbol (S) remains. The stored phrases are actually rules, because there is a possibility of reconstructing the original syntax tree with their help.

3.3 Tree disassembly with tree shapes

The disadvantage of one-level context-free rules is that important information disappears from the contextual conditions of rule applications. This can cause errors

Syntax tree of a short sentence:



Bracketed sentence:
($_S$($_{CP}$($_{NP}$Np-sn|Mihály) ($_{VP}$Vmis3s---n|mosolyog)PUNCT|.))

One level patterns:
($_{NP}$Np-sn|Mihály)
($_{VP}$Vmis3s---n|mosolyog)
($_{CP}$ NP VP PUNCT|.)
($_S$ CP)

Retrieved grammar:
NP → Np-sn|Mihály
VP → Vmis3s---n|mosolyog
CP → NP VP PUNCT|.
S → CP

Figure 1: Short example for tree disassembling (Mihály is smiling.)

in syntax parsing especially in the case of short rules. For example in Figure 1 there is a short rule: *NP → Np-sn|Mihály.* In a more complex sentence (e.g. *the friendly Mihály is smiling*) the application of this rule causes an error because the noun phrase consists of not only Mihály (but *the friendly Mihály*). For this rule we need a context to know whether its application is error-free. This derivation may not be completed with an S symbol and in this case the derivation is dropped, but due to short context-free rules there is a chance that if we take longer sentences many false derivations will appears in the derivation forest, making the runtime longer and the selection of the best tree harder.

Our solution for reducing problems with context-free rules is the extraction of *subtree patterns* or simply *tree patterns.* These patterns contain more than one node in hierarchical structures i.e. they are trees. The description of tree patterns includes word positions with their stems and lexical codes and bracketed phrases. Applying tree patterns instead of one-level context-free rules, short phrases are placed together with their contexts, so the derivation with tree patterns reduces ambiguities and raises the accuracy score. But using patterns brings another question: what size of trees are useful? Big trees are too wide or too deep, or both, and

too specific. Grammars with over-specific rules can also result in poor accuracy scores on unknown texts because of missing covering rules. Hence not only short rules must be eliminated, but big trees as well.

To determine the proper size of trees extracted from a training corpus *tree shape types* are used. Tree shape types are a general form of subtrees that are defined with some property in their description. These types were chosen by linguistic experts studying which syntactic structures appear most often in annotated sentences. In the end the following two tree shape types were found:

1. "Hole":
   - Contains at least two terminals
   - It can be divided into one deepen and one risen parts
   - It has a recursive structure in Hungarian, the inside tree appended or prefixed with terminals
   - Example (Eng. 'on the edge of a terrible whirlpool'):

```
(NP
    (NP
        Ti|egy
            (ADJP
                Afp-sn|rettenetes
            )
        Nc-sn|örvény
    )
    Nc-sp—s3|szélén
)
```

2. "String":
   - Its depth is 2
   - It may contains more subtrees, terminals or substituted nodes
   - The depth of its subtrees is 1
   - It can be enumerated in Hungarian, the small trees, terminals or substituted nodes follow each other and these are included by a single phrase
   - Example (Mihály és Erzsi, 'Michael and Liz'):

```
(NP
    (NP
        Np-sn|Mihály
    )
    (C
        Ccsw|és
    )
    (NP
        Np-sn|Erzsi
    )
)
```

4 Learning tree patterns

In this section the learning task of syntactic tree patterns are described, which contains the preprocessing of training data, and the generalization and specialization of tree patterns. An improved version of the RGLearn algorithm [6] called PGS (*Pattern Generalization and Specialization*) was used as a tree pattern learner. The novelty of PGS is the use of λ parameters which have an influence on the quality of learned tree patterns. The pattern unification method and the search method for the best patterns were also modified.

4.1 Preprocessing of training data

The initial step for generating training data is to collect syntactic tree patterns from an annotated training corpus. The complete syntax tree of a sentence must be divided into separate trees and a cascade of tree building rules to help prepare the parser in the reconstruction of it. In parsing, using context-free grammars has a lot of advantages, but the conditions of pattern usage may completely disappear. Some structural information can be retained if tree patterns are used. To generate cascaded grammar, linguistic experts have defined the following processing levels for the Hungarian language:

- Short tree patterns of nominal, adjectival, adverbial and pronominal phrases.
- Recursive patterns of nominal, adjectival, adverbial and pronominal phrases.
- Recursive patterns of verb phrases.
- Recursive patterns of sub-sentences.

4.2 The generalization of tree patterns

Using the collected tree patterns the syntactic parser is able to reconstruct the tree structure of training sentences. In order to perform the syntactic parsing of an unknown text to a fair accuracy, the collected tree patterns must, however, be generalized. Generalization means that the lexical attributes of each tag are neglected except for the POS codes. In this phase the learning problem is transformed into a classification problem. Namely the following question should be answered: which set of lexical attributes provides the best result for the decision problem of tree pattern matching i.e. a given tree structure covers a given example or not. To support the learner, positive and negative examples are collected from training sets for each tree type. The example in Figure 2 shows the complete tree pattern learning process.

4.3 Specialization of tree patterns

Negative examples are bad classifications of generalized tree patterns and they must be eliminated. Therefore specialization selects each possible lexical attribute from

Sentence parts (examples):
1: . . . $(_{NP}Tf(_{ADJP}Afp - sn)Np - sn)$ . . .
2: . . . $(_{NP}Tf(_{NP}Afp - sn)Nc - pa)$ . . .
3: . . . $(_{NP}Ti(_{NP}Afs - sn))(_{NP}Nc - s2)$ . . .
4: . . . $(_{NP}Tf(_{ADJP}Afp - sn)Nc - sn)$ . . .
5: . . . $(_{NP}Tf(_{ADJP}Afp - sn)(_{ADJP}Afp - sn))$ . . .

One of four possible generalized pattern: $(_{NP}T * (_{ADJP}A*)N*)$
Coverage: positive $\{1, 4\}$, negative $\{2, 3\}$, uncovered $\{5\}$
Specialized pattern: $(_{NP}T * (_{ADJP}A*)N???n)$
Coverage: positive $\{1, 4\}$, negative $\{\}$, uncovered $\{2, 3, 5\}$

Notations:
The first letter of morpho-syntactic codes is the part of speech
Determiner: T*, Adjective: A*, Noun: N*
A letter of any kind: ?, One or more letters of any kind: *

Figure 2: A tree pattern learning example.

positive examples making new tree patterns, and tries to find the best tree patterns via unification.

The initial step of specialization generates all possible new tree patterns by extending generalized tree patterns with exactly one attribute from the covered positive examples. The next steps of specialization extend the set of tree patterns with all possible new tree patterns by a combination of each pair of tree patterns. The combination of two tree patterns means the union of their lexical attributes. To avoid the exponential growth of a tree pattern set weak tree patterns are excluded by applying error statistics on positive and negative examples. The following score of a given tree pattern is used as the target for maximization:

$$score = \lambda_1 * (pos - neg)/pos + \lambda_2 * (pos - neg)/(pos + neg)$$

where *pos* is the number of covered positive examples, *neg* is the number of covered negative examples and $\lambda_1 + \lambda_2 = 1$.

Fruitful unifications dramatically decrease the negative coverage. The score maximization operates in parallel on every positive example. A new tree pattern is stored only if a covered positive example exists where the score of the new tree pattern is greater than the previous maximum value. Specialization ends when the current step did not improve any maximum value.

Appropriate setting of λ factors in linear combination can provide the optimal tree pattern set. A greater λ_1 may result in tree patterns with high coverage, while a greater λ_2 may resulting a higher accuracy but there is a possibility of low tree patterns appearing with a low coverage.

4.4 Producing probabilistic grammar

The syntax tree of a sentence can generally be derived in different ways, especially when using a large grammar. The choice of a best derivation from the derivation forest requires additional information. One of the possible ways of doing this is a making of PCFG (Probability Context Free Grammar). Using a PCFG, the probability of a derivation is the product of probabilities of the applied rules, but the product may be replaced by other operators, e.g. a max operator. After evaluating each derivation, the derivation with a higher probability will selected. It is described with the following formulas:

$$P(Derivation) = \prod_{T \to \beta \in Derivation} P(T \to \beta | T) \quad (1)$$

$$Best = \underset{Derivation \in Forest}{\operatorname{argmax}} P(Derivation) \quad (2)$$

The conditional probability of a rule is computed with the following formula of Maximum Likelihood Estimation:

$$P(T \to \beta | T) = \frac{Count(T \to \beta)}{Count(T)} \quad (3)$$

Finally, the last formula shows that the conversion of a CFG to a PCFG is based on rule application statistics on the training data, namely, counting the proper coverage of each rule and counting the node labels in the annotated syntax trees. This method is applicable for tree patterns as well. Counting the coverage of tree patterns is the same as counting coverage of one level rules. The left symbol for a CFG rule of tree patterns is a root of the subtree:

Tree patterns:

$(_{NP}(_{NP}$ T* $(_{ADJP}$ A*) N??s?) N???????s3)

$(_{NP}(_{NP}$ N* $(_{C}$ C*) N*))

Rule form (brackets indicate the structural information):

NP $\to$ $(_{NP}$ T* $(_{ADJP}$ A*) N??s?) N???????s3

NP $\to$ $(_{NP}$ N* $(_{C}$ C*) N*)

5 Syntactic parsing

The main task of the syntactic parser is to find the most likely parse tree for input sentences. Input data contains disambiguated POS tag labels and it may come from the *Szeged Corpus*during the testing phase of the method, or it may be provided by a POS tagger tool [9] as a practical application of the method. There are natural language processing modules used to determine various linguistic features for syntactic parsing when the process starts with plain text: tokenization, sentence segmentation, morpho-syntactic analysis and part-of-speech (POS) tagging.

Parsing with a PCFG can be done in polynomial time - but disambiguation - namely the selection of best possible parse tree from the parse forest, is an NP-hard

Figure 3: Two different subderivations for the same part of a sentence.

problem. Sima'an [13] demonstrated that there is no deterministic polynomial time algorithm for finding the most probable parse of a sentence. Owing to this fact, it is not efficient if the parser determines all possible derivations with probabilities, and then selects the best one. Hence, the Viterbi algorithm [18] is applied in our parser to find the most probable derivation, because it can perform it in cubic time. The basic idea behind the Viterbi approach is the elimination of low probability subderivations in a bottom-up fashion. Two different subderivations for the same part of sentence and with the same root can both be included by derivations in the same way. Thus, if one of these two subderivations has a lower probability, it will be eliminated. The selection situation is illustrated by a short example in Figure 3.

The Viterbi elimination method is applicable for higher levels in bottom-up parsing and finally for the selection among S roots of the derivation forest. There are a lot of low probability subderivations that are pruned through parsing to speed up the process. The syntactic parsing of a sentence involves the following:

```
Take a POS tagged sentence
    BEST_TREE = nil
    RULE_LEVEL = 0
    repeat
        derive with each tree patterns from Cascade_Set ( RULE_LEVEL )
        foreach NEW_NODE do // Viterbi elimination for new nodes
            if exist SAME_NODE as NEW_NODE then
                Eliminate_Weaker ( NEW_NODE , SAME_NODE )
        if exist ROOT_NODE == S then
            BEST_TREE = Tree ( ROOT_NODE )
        RULE_LEVEL = RULE_LEVEL + 1
    until (applicable tree pattern exist)
```

Note: depending to the rule set used, the full parse tree with an S root does not always exist for an arbitrary input sentence.

Table 1: Test results on the two corpus domains.

Text category	Precision	Recall	$F_{\beta=1}$
Corpus version 1.0	82.27%	79.35%	80.78%
Business news extension	85.83%	81.46%	83.59%

6 Experiments

The training and evaluation datasets were taken from various parts of the Szeged Corpus. One of the domains we examined was the first version of the corpus containing texts from five different topic areas and the second domain was the business news extension of the Szeged Corpus. The training ad test was performed using 10-fold cross-validation. The sentences of domains were randomly divided into ten parts. Then ten different train (90%) and test (10%) sets were pieced together from parts and learning and testing was performed ten times. The average of the results on ten test sets is shown in Table 1.

In general, the precision score is higher than the recall, because of the unknown structures. The better results in business news extension may be caused by the different characteristics of domains. The business news part contains relatively homogeneous texts with often repeated phrases and idioms. The first domain is more heterogeneous in its five topics. Based on the experiments, the syntactic parser - as an element of a natural language processing pipeline - provided enough information for information extraction.

7 Summary and future work

In this paper a general tree pattern learning method for syntactic parsing was presented. The initial grammar for learner was collected from a large corpus domain using tree shapes to disassemble a syntactic tree of annotated sentences. The PGS algorithm presented, an improved version of the RGLearn machine learning algorithm, performs generalization and specialization to produce grammar. Probability values of learned tree patterns were also computed. For the parsing algorithm, Viterbi searching was used to speed up finding the most probable derivation. Based on results the syntactic parser, as an element of a natural language processing pipeline, provides sufficiently rich information to support information extraction.

In the future, in parallel to the development of the Szeged Corpus, the developing machine learning methods are intended for the same problems as syntactic parsing i.e. named entity recognition and semantic frame identification. Improving the results of syntactic parser building in other methods, e.g. system combinations and the usage of ontological information that could be the extension of a morpho-

syntactic description is also planned. The system described here will be primarily applied to the business news domain. In the 6th Framework Programme of the European Commission, an international research consortium is planning to develop a multilingual IE system for the above-mentioned two domains.

References

[1] Abney, S. Partial parsing via finite-state cascades. In *Proceedings of ESSLLI'96 Robust Parsing Workshop*, pages 1–8, 1996.

[2] Abney, S. *Parsing by chunks.* Kluwer Academic Publishers, 2003.

[3] Alexin, Z., Csirik, J., Gyimóthy, T., Bibok, K., Hatvani, Cs., Prószéky, G., and Tihanyi, L. Manually annotated hungarian corpus. In *Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics EACL03*, pages 53–56. Budapest, Hungary, 2003.

[4] Argamon, S., Dagan, I., and Krymolowski, Y. A memory-based approach to learning shallow natural language patterns. In *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 67–73. Montreal, 1998.

[5] Erjavec, T. and Monachini, M., ed. *Specification and Notation for Lexicon Encoding.* Copernicus project 106 MULTEXT-EAST; Work Package WP1 - Task 1.1 Deliverable D1.1F., (1997.

[6] Hócza, A. Noun phrase recognition with tree patterns. *Acta Cybernetica, vol 16*, pages 611–623, 2004.

[7] K., Simov. Clark - an xml-based system for corpora development. In *Proceedings of the Corpus Linguistics 2001 Conference*, pages 553–560. Lancaster, 2001.

[8] Kis, B., Naszódy, M., and Prószéky, G. Complex hungarian syntactic parser system. In *Proceedings of the MSZNY 2003*, pages 145–151. Szeged, Hungary, 2003.

[9] Kuba, A., Bakota, T., Hócza, A., and Oravecz, Cs. Comparing different pos-tagging techniques for hungarian. In *Proceedings of the MSZNY 2003*, pages 16–23. Szeged, Hungary, 2003.

[10] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: the penn treebank. *Association for Computational Linguistics*, 1993.

[11] Prószéky, G. and Kis, B. A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 261–268. College Park, Maryland, USA, 1999.

[12] Ramshaw, L. A. and Marcus, M. P. Text chunking using transformational-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*. Association for Computational Linguistics, 1995.

[13] Sima'an, K. Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proceedings of COLING-96*. Copenhagen, 1999.

[14] Tjong Kim Sang, E. F. Noun phrase recognition by system combination. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 50–55. Seattle, 2000.

[15] Tjong Kim Sang, E. F. and Veenstra, J. Representing text chunks. In *Proceedings of EACL '99 Conference*. Association for Computational Linguistics, 1999.

[16] Váradi, T. The hungarian national corpus. In *Proceedings of the Second International Conference on Language Resources and Evaluation LREC2002*, pages 385–389. Las Palmas de Gran Canaria, 2002.

[17] Váradi, T. Shallow parsing of hungarian business news. In *Proceedings of the Corpus Linguistics 2003 Conference*, pages 845–851. Lancaster, 2003.

[18] Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans Information Theory, vol IT-13*, pages 260–269, 1967.

CONTENTS

Selected Papers from the *2nd Conference on Hungarian Computational Linguistics*

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János